大数据系列丛书



大数据挖掘及应用

王国胤　刘群　于洪　曾宪华　编著

清华大学出版社

大数据系列丛书

大数据挖掘及应用

王国胤　刘　群　于　洪　曾宪华　编著

清华大学出版社
北　京

内 容 简 介

本书围绕大数据背景下的数据挖掘及应用问题，从大数据挖掘的基本概念入手，由浅入深、循序渐进地介绍了大数据挖掘分析过程中的数据准备和预处理方法、数据可视化技术、数据挖掘理论和经典算法、常用大数据分析计算平台的编程模型、并行化程序设计技术、统计分析R语言基础等内容。其中数据挖掘理论和经典算法不仅覆盖了传统的关联分析、分类和聚类，还包括深度学习理论等数据挖掘研究和发展的潮流主题。每一章内容都尽量从不同角度进行深入浅出的剖析，还配以丰富的习题和参考文献，对于读者掌握大数据挖掘及应用领域的基本知识和进一步研究都具有参考价值。本书可以作为高校本科相关专业数据分析类课程教材和面向各专业的数据科学通识教材，也可供广大IT从业人员参考。

图书在版编目(CIP)数据

大数据挖掘及应用/王国胤等编著. —北京：清华大学出版社，2017(2017.12重印)
(大数据系列丛书)
ISBN 978-7-302-46927-8

Ⅰ. ①大…　Ⅱ. ①王…　Ⅲ. ①数据采集－研究　Ⅳ. ①TP274

中国版本图书馆CIP数据核字(2017)第074345号

责任编辑：张　玥　战晓雷
封面设计：常雪影
责任校对：焦丽丽
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：26　　**彩　插**：8　　**字　　数**：624千字
版　　次：2017年7月第1版　　**印　　次**：2017年12月第2次印刷
印　　数：2001～3500
定　　价：59.50元

产品编号：074001-02

序　言

PREFACE

人类社会，在经历过农业社会、工业社会之后，经过近几十年的信息化建设，已经发展成为一个信息化的现代社会。人类不但具备了数据信息的全面感知、快速传输、高效计算、海量存储等方面的超强能力，而且信息化也已经深刻而广泛地影响到社会的各行业领域和生产生活的方方面面。在数据信息的全面感知获取方面，从深空到深海和深地的探测感知、从企业集团到生产设备的监控感知、从智慧城市到个人信息的监测感知……人类已经从各个角度、多个层次进行了数据信息的丰富感知采集。无所不在的物联网和传感器，使得数据信息呈"爆炸式增长"。在信息快速传输方面，当今的骨干网络带宽进入了TB(PB)/秒级，接入网带宽进入了MB(GB)/秒级，信息传输的便捷程度前所未有。在高效计算能力方面，单台高性能计算机系统的浮点运算速度已超过亿亿次/秒，便携式计算机的运算处理速度也在日新月异地发展变化。在数据信息的海量存储方面，无论是便携式存储器，还是云存储系统，数据存储的代价不断迅速下降，容量快速上升。随着计算机和信息网络技术的飞速发展，以及信息化技术与其他行业领域技术的交叉融合，形成了"互联网＋"新业态，数据成为了未来社会发展的新资源。如何通过大数据挖掘，有效实现数据这种新资源的社会价值，成为社会各行业领域关注的一个核心问题，甚至诞生了"数据科学与工程"这样一个新兴的学科和专业。大数据的挖掘与分析应用技术，最近被列入到《国务院关于积极推进"互联网＋"行动的指导意见》《促进大数据发展行动纲要》和《"十三五"国家战略性新兴产业发展规划》等多个国家战略规划的文件之中，国务院2017年7月发布的《新一代人工智能发展规划》也明确将大数据智能列为一个重要发展方向。培养学生在大数据工程应用方面的能力，提升学生对未来社会信息化和智能化建设的适应能力，成为社会对人才培养的一大紧迫需求。

重庆邮电大学大数据课程组在大数据工程技术研究与教学实践中，经过几年的探索，积累了一些经验。课程组的几位老师合力编写了这本《大数据挖掘及应用》教材，希望能够帮助对大数据挖掘技术感兴趣的读者了解学习相关的理论知识和技术方法，助推大数据工程技术人才的培养。在教材的教学内容设计和知识结构组织中，紧密围绕各行业领域推进"互联网＋"模式的信息化建设对大数据工程技术人才的需求，循序渐进地安排了从数据的准备与预处理，到数据的可视化与关联分析，再到数据的分类、聚类和深度学习等大数据挖掘处理的主要基础理论知识和模型技术方法，以及从R语言到Hadoop系统和并行化计算等大数据挖掘处理的主流工具与平台技术，特别是通过丰富的教学案例实现了理论学习与实践提高的有机融合。因此，本书适合各高校信息类和管理类本科专业的课程教学，对其他理工科专业学生以及从事信息化和智能化建设的各行业领域工程技

术人员也具有重要的参考价值。

为了便于本书的教学组织，课程组准备了配套的教学课件和电子资源等辅助教学资料。课程组建设的“大数据分析与处理”在线开放课程（课程网站：http://www.cqooc.com（面向重庆市高校学生）、http://cqupt.gaoxiaobang.com（面向全国读者）），可以作为本书的网络在线学习参考资源，供自学的读者参考学习，采用本书教学的教师也可将其作为参考资料。

在本书即将重印之际，要特别感谢参与编写和相关教学资源建设的各位老师！由于你们的不辞辛劳与通力合作，才有今天教材的完稿出版，并得到读者的厚爱。本教材的编写得益于重庆邮电大学大数据课程组全体老师、计算机软件教学部国家级教学团队和计算智能重庆市重点实验室的众多老师和学生的支持，他们在大数据方面长期的教学和科研积累为教材的撰写奠定了坚实的基础。感谢清华大学出版社对本书策划和出版的大力支持。在此，对大家的大力支持、辛勤工作与热情奉献，表示诚挚的谢意！

课程组还将不断总结科研成果和教学经验，吸取广大师生和读者的意见，进一步将本书锤炼成为一本精品教材。

重庆邮电大学　王国胤

写于　2017年10月10日

前　言

PREFACE

今天,“大数据”已经成为一个非常时尚的概念,得到广泛应用,不仅受到 IT 从业人员的重视,而且影响到了自然科学、社会科学、人文科学等领域的广大从业者,并对社会经济的各行业产生了深远的影响。大数据已经不再是对大量数据的处理问题了,最重要的是对大数据进行分析,只有通过分析才能从数据中获取深入的、智能的、有价值的信息与知识。不断增长的大数据呈现出数据量大、种类繁多、增速很快以及隐藏价值大的特点,因此好的分析技术和方法在大数据应用领域显得尤为重要。本书围绕大数据背景下的数据挖掘和应用问题,从大数据挖掘的基本概念入手,由浅入深、循序渐进地介绍了大数据挖掘分析过程中的数据准备和预处理技术、数据可视化技术、数据挖掘的基本方法、大数据分析计算的常用平台架构编程方法、并行化程序设计技术以及常用的 SPSS 统计分析工具、流行的统计分析 R 语言等内容。本书不仅面向在校大学生,而且面向社会广大的 IT 从业人员,有助于读者了解大数据挖掘所涉及的基本技术和方法。作者力图使读者通过学习,提高数据分析的实践动手能力,拓展在数据分析领域的视野。

参与编写本书的所有作者均来自重庆邮电大学计算智能重庆市重点实验室,都具有多年从事数据挖掘、机器学习等人工智能领域的科研和教学实践经验。本书在结构设计与内容安排上既体现了所有作者的群体智慧,也体现了本领域的近期发展和前沿成果。

目前,大数据的知识挖掘及应用方法逐渐成为各高校信息类和管理类本科专业的必修课程内容,同时,作为面向各专业的通识课也广受欢迎。本书作为立足于本科教学的教材,具有如下特色:

(1) 在逻辑安排上循序渐进,由浅入深,便于读者系统学习。

(2) 内容丰富,信息量大,融入了大量本领域的新知识和新方法。

(3) 作为教材,在每一个环节都配有与理论学习内容相结合的案例分析,不仅有学生参赛作品展示,还有采用 Python 和 R 语言编写的应用实例,尤其是在第 10 章还给出了完整的大数据分析应用实际案例,使读者能够在大数据平台上实际感受一个完整的数据分析过程。

(4) 图文并茂,形式生动,可读性强。

全书内容分为 3 部分,共 11 章。第 1 部分是数据挖掘及应用导论,由第 1～3 章组成。

第 1 章主要是关于大数据挖掘及应用的概论。本章讨论了大数据挖掘及应用普及的发展历程及重要性,探讨了目前所面临的挑战和问题,介绍了数据挖掘的基本概念、功能和方法,进而对大数据挖掘的计算框架和处理流程进行了分析总结。在本章教学中,可以紧跟最新事件,以生动的实例、动画、视频等形式激发学生兴趣。建议 2 学时。

第 2 章的主题是数据认知和数据准备。本章首先从数据分析的定义和流程入手，给出了评价高质量数据的指标。然后对数据由什么类型的属性或字段组成，每个属性具有何种类型的数据值，是离散属性还是连续属性进行了描述，进而讨论了数据的中心趋势和离散趋势度量指标，以及数据相似性和相关性的计算方法，并着重探讨了数据预处理中数据清理、数据集成、数据归约和数据变换的技术。最后简述了目前常用的数据统计分析和预处理工具，并用一个案例对 SPSS 工具进行了介绍。在本章教学中，可以以一种数据统计分析工具为背景，进行形象具体的介绍。建议 6 学时。

第 3 章主要介绍数据可视化技术。本章从可视化技术的应用开始，介绍了最常用的高维数据可视化方法和网络数据可视化方法，最后通过两个竞赛案例对可视化技术的实际应用作了详细的讲解。可视化技术能够以图形的表现方式帮助人们识别隐藏在杂乱数据集中的关系、趋势和偏差等有价值信息。在本章教学中，可以运用可视化软件进行案例演示，激发学生的学习兴趣。建议 6 学时。

第 2 部分是数据挖掘及应用的方法论，由第 4～8 章组成。

第 4 章包含数据关联分析的基本知识和主要经典算法。数据关联分析是数据挖掘中应用最早和最成熟的一类方法。本章从一个问题案例出发，先后介绍了关联规则分析的基本概念以及 3 种典型的频繁项集挖掘算法，并对关联规则的有效性进行了探讨，将学习内容扩展到频繁项集挖掘的一些高级方法，例如多维关联规则挖掘和多层关联规则挖掘，最后使用 Python 语言给出了经典 Apriori 算法的一个应用案例。在本章教学中，可以结合数据挖掘领域著名的开源软件 weka 进行演示教学，让学生形象地体会经典关联分析算法产生的效果和使用全过程。建议 6 学时。

第 5 章包含数据分类分析的基本知识以及主要经典算法。本章从介绍分类的基本概念入手，讲解了数据分类分析的基本方法，包括最常用的决策树分类器，基于概率统计思想的贝叶斯分类算法，具有统计学习理论坚实基础的支持向量机算法，以及通过构建一组基于学习器进行集成学习的 Adaboost 算法，最后使用 Python 语言给出了一个具体案例，使读者能够熟悉数据分类分析的全过程。在本章教学中，可以结合数据挖掘领域著名的开源软件 weka 进行演示教学，让学生形象地体会经典分类算法产生的效果和使用全过程。建议 6 学时。

第 6 章包含数据聚类分析的基本知识和主要经典算法。本章首先引入聚类的基本概念，进而讲解各种聚类算法，包括基于划分的 k-means 算法和 k 中心点算法，基于层次的算法，基于密度的 DBSCAN 算法以及基于概率模型的期望最大化算法，并简要讨论了评估聚类方法的准则，最后使用 Python 语言给出一个案例，帮助读者更好地理解聚类分析技术。在本章教学中，可以结合数据挖掘领域著名的开源软件 weka 进行演示教学，让学生形象地体会经典聚类算法产生的效果和使用全过程。建议 4～5 学时。

第 7 章探讨人工智能研究中的一个重要的新领域——深度学习。本章首先介绍深度学习的发展和基本概念，然后具体分析了深度学习的几种经典模型与算法，包括最常用的深信网、深玻尔兹曼机、栈式自动编码器和卷积神经网络，最后介绍了几种深度学习开源模型并给出了一个具体案例，帮助读者了解深度学习在实际应用中的完整工作过程。在本章教学中，可以结合书中介绍的某一种深度学习开源框架，采用相应的数据集进行演示

教学，让学生形象地体会深度学习算法产生的效果和使用全过程。建议6学时。

第8章介绍目前流行的统计分析R语言。本章首先从R语言的下载安装开始，介绍R语言的基本技术，包括运行方法、常用操作、包的使用、常用数据结构、编程结构以及与数据挖掘和图形绘制相关的包，最后使用R语言给出了一个从数据预处理到数据分析的具体案例，使读者能够熟悉使用R语言做数据分析的全过程。在本章教学中，可以通过类似实训课程或者视频录像的形式，让学生形象地体会并掌握R语言的操作方法和编程基础。建议4学时。

第3部分属于数据挖掘及应用的进阶部分，由第9～11章组成。

第9章的主题是大数据分布式存储与并行计算的平台Apache Hadoop及其编程框架。本章从介绍Hadoop集群的基本概念开始，讲解了HDFS基本操作、MapReduce并行计算基础、基于Storm的分布式实时计算以及基于Spark Streaming的分布式实时计算等内容。在各节中都给出了若干案例，以供读者在实际编程过程中进行参考。在本章教学中，可以要求学生紧扣教材，完成各节中的案例，增强身临其境的体验。建议4～5学时。

第10章介绍大数据分析处理算法的并行化基础理论和技术。本章介绍了并行计算算法的基本概念，以MR-KMeans算法为典型案例分析了其在MapReduce计算框架下的并行化，并基于Mahout和MLlib对该算法进行了并行化实现，最后给出了3个完整的MapReduce平台下数据分析的具体案例，使读者能够了解在大数据平台上进行数据分析的全过程。在本章教学中，可以通过专门的视频演示，让学生理解并行化编程的复杂实际操作。建议4～5学时。

第11章主要关注大数据挖掘及应用的发展趋势和研究前沿。本章首先从大数据时代发展的回顾与展望开始，介绍了大数据发展过程中出现的典型新数据类型以及新挖掘分析方法，并在最后对大数据的发展进行了展望。建议2学时。

本书各章提供的教学建议和学时安排仅供教师参考。教师可以根据教学过程中的实际安排删减内容和调整学时。本书还提供了一些丰富的教学资源供教师教学参考和学生学习时使用。以本书各章内容为基础的在线微视频开放课程已经在cqupt.gaoxiaobang.com网站上线，教师可以通过课前推送的方式，指导学生观看相关视频进行课前预习，其他读者可以通过该视频课程巩固和完善对各个知识点的理解。除此之外，我们还提供了一些其他的附加材料，包括每章的幻灯片、每章涉及的案例的软件程序、课后习题解答以及一些案例的演示视频，以上这些资料在清华大学出版社的网站上向教师提供。

本书的第1、11章由王国胤和张旭编写，第2、4章由刘群编写，第3章由秦红星编写，第5、6章由于洪编写，第7章由曾宪华编写，第8章由吴思远编写，第9章由李智星编写，第10章由张旭编写。全书架构由王国胤负责设计，王国胤和刘群负责统稿。

本书的编写得到了重庆邮电大学计算智能重庆市重点实验室和计算机科学与技术学院教师们的大力支持和帮助，也得到了许多研究生的支持，他们帮助收集并整理了大量资料。没有他们的帮助，本书很难在约定时间内完成。在此，感谢他们对本书的写作所做出的各种贡献。

限于作者学识和经验，书中难免会出现不足和遗漏之处，欢迎读者指出，一旦问题被证实，我们将给出更新勘误表，并对您表示感谢。评论和建议请发往 liuqun@cqupt.edu.cn，我们很高兴能听到您的声音。

感谢读者的鼎力支持，本书得以再次印刷。由于作者学识浅显，经验有限，书中出现了一些遗漏和错误，已经进行了更正。欢迎读者继续指出本书中的错误，一经确认，我们将更新勘误表，并对您的贡献致谢。评论和建议请发往 liuqun@cqupt.edu.cn，我们很高兴能听到您的声音。

作　者

2017 年 11 月

目 录

CONTENTS



大数据挖掘及应用概论

数据挖掘(data mining)是数据库知识发现(Knowledge-Discovery in Databases,KDD)中的一个步骤。其一般是指从大量的数据中通过算法搜索隐藏于其中的信息的过程。随着信息科技的进步和网络的发达、计算机运算能力的增强以及数据存储技术的不断改进,人类社会正迈向信息时代,数据的爆炸式增长、广泛可用和巨大体量使得我们的时代成为真正的数据时代,迫切需要功能强大和通用的工具,以便从大数据中发现有价值的信息,并将这些数据转换成有用的信息和知识,所获取的信息和知识可以广泛用于各种应用,包括商务管理、生产控制、市场分析、工程设计和科学探索等。数据挖掘方法利用了来自如下一些领域的技术思想,如来自统计学的抽样、估计和假设检验,来自人工智能、模式识别和机器学习的搜索算法、建模技术和学习理论,以及来自包括最优化、进化计算、信息论、信号处理、可视化和信息检索等的重要支撑。随着数据量越来越大,源于高性能分布式并行计算和存储的技术在大数据挖掘和应用中显得越来越重要。

本章是一个概论,试图较完整地展现在大数据挖掘和分析中对各种各样的应用进行有效数据模式发现的方法,特别是那些开发有效的、可扩展的大数据分析工具和分布式并行计算平台的卓越技术。本章组织结构如下:首先介绍大数据智能分析处理的普及和应用的概貌(1.1 节),接下来探讨大数据的发展及挑战问题 (1.2 节),概述数据挖掘(1.3 节),介绍大数据分析处理和挖掘的框架(1.4 节),最后对大数据时代“互联网+”的未来发展进行展望分析,提出智能互联的概念(1.5 节)。

1.1 大数据智能分析处理的普及和应用

本节给出云计算及大数据智能分析处理的基本概念及其应用术语,为读者研究云计算和大数据建立基础。1.1.1 节和 1.1.2 节分别介绍云计算和大数据的基本概念,1.1.3 节介绍云计算和大数据的智能应用。

1.1.1 云计算

2015 年 1 月,国务院发布了《关于促进云计算创新发展培育信息产业新业态的意见》,在这个文件中为云计算和大数据分析等关键技术领域描绘了未来的发展蓝图。云计算是人机交互的互联计算,大数据将人类社会带入了物理空间、社会空间与数据空间共生的三元空间世界。“互联网+”正向智能的深度互联发展,大量的数据资源有待利用和价

值发现。LinkedIn 于 2014 年发布了最新的调研分析，称最大的求职法宝是统计分析与数据挖掘技能。美国招聘网站 Glassdoor 也有一个报告，称程序员的平均年薪只有 6.5 万美元，数据科学家却高达 11.9 万美元。希望读者能够通过对本书的学习，拓宽职业道路，开拓研究视野，提升年薪，增强发展能力。

首先来看看人们对大数据、云计算的关注度变化(数据来自百度搜索指数①)。该数据展示了互联网用户对关键词搜索的关注程度及持续变化情况(算法说明：以网民在百度的搜索量为数据基础，以关键词为统计对象，科学分析并计算出各个关键词在百度网页搜索中搜索频次的加权)。根据数据来源的不同，搜索指数分为 PC 搜索指数和移动搜索指数。本章摘录的是 2011—2016 年的整体趋势，如图 1-1 所示。可以看出，“云计算”自 2008 年引起关注，之后的搜索量迅猛增长并在 2011 年底呈现较高的关注热度，且一直维持在一个较高的搜索水平。2016 年底，随着全球首款亿级并发云服务器系统在我国量产，全球云计算基础硬件设施进入一个全新的时代，通用服务器并发从“万”级步入“亿”级②，人们对云计算的关注度再次出现了井喷式增长。“大数据”这个词虽然从 2012 年才受到人们的关注，但是在很短的时间内其搜索量就超越了“云计算”，之后一直呈现较高的热度。

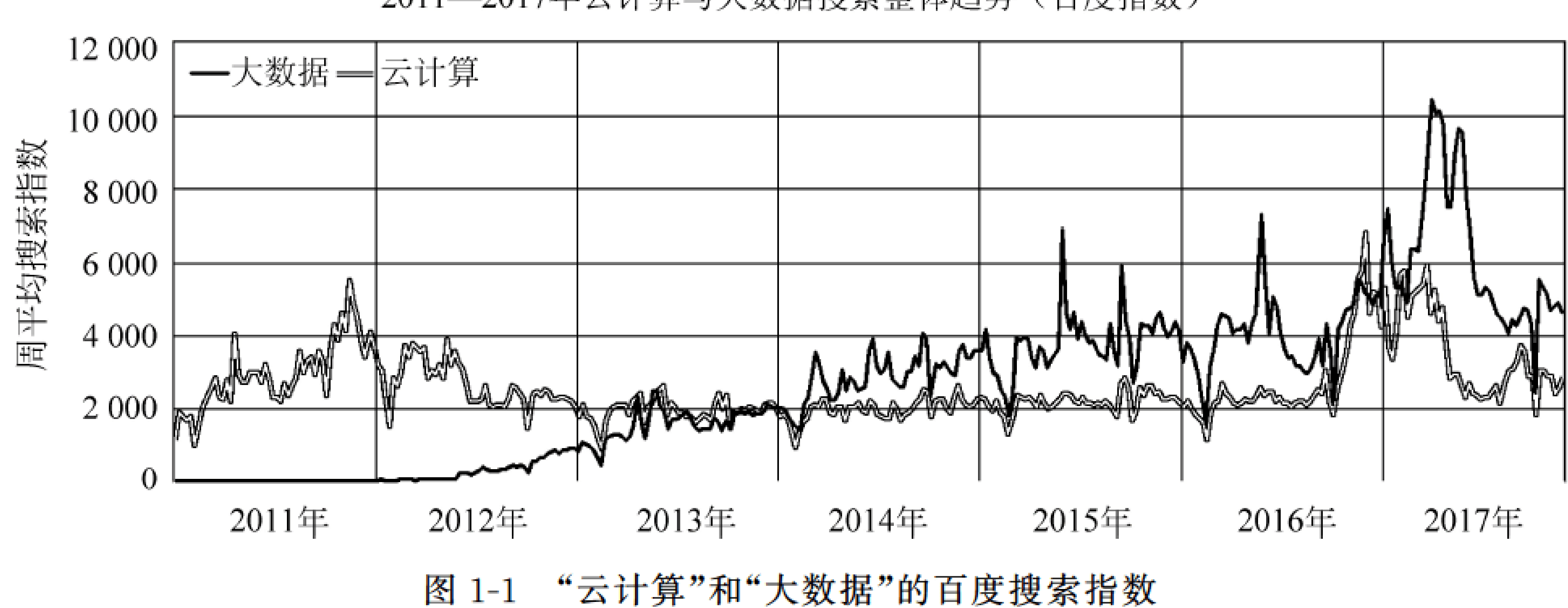


图 1-1 “云计算”和“大数据”的百度搜索指数

什么是云计算呢？先看一个简单的例子。我们每天都要用电，但不是每家自备发电机，它由电厂集中提供；我们每天都要用自来水，但不是每家都有井，它由自来水厂集中提供；开车需要加油，但不是每家都有加油站；做饭需要肉蛋禽菜，但不是每家都会养猪种菜。我们从国家、社会或者第三方团体采购或享受这些服务，这种模式极大地节约了资源，方便了人们的生活。面对众多独立运行的计算机，可不可以像使用水和电一样统一使用计算机资源，而不再单独建设、购买和使用呢？这些想法最终导致了云计算的产生。2006 年谷歌推出了“Google 101 计划”，并正式提出“云”的概念和理论。随后亚马逊、微软、惠普、雅虎、英特尔、IBM 等公司都宣布了自己的“云计划”。云安全、云存储、内部云、外部云、公共云、私有云……一堆让人眼花缭乱的概念不断冲击着人们的眼球。

---

① 百度指数(https://index.baidu.com/)：截至 2017 年 11 月 22 日“云计算”和“大数据”搜索指数。

② 科技部. 中国首款亿级并发云服务器系统正式量产. 2016-12-30. http://www.most.gov.cn/kjbgz/201612/t20161230_130074.htm.

那么，如果要给云计算一个严格的定义，又应该是什么呢？

- 政界定义：云计算是推动信息技术能力实现按需供给、促进信息技术和数据资源充分利用的全新业态，是信息化发展的重大变革和必然趋势。(《国务院关于促进云计算创新发展培育信息产业新业态的意见》国发[2015]5号①)
- 学界定义：云计算是基于互联网的相关服务的增加、使用和交付模式，通常涉及通过互联网来提供动态易扩展且经常是虚拟化的资源。(百度百科②)
- 本书观点：云计算是人机交互的互联计算系统。

云计算不是简单地把一大堆计算机通过互联网连接起来。它彻底改变了人类的生活方式、工作方式和休闲方式，改变了社会的政治、经济、教育、商务、健康与娱乐机制，已经成为推动技术发明和社会变革的最强大的发动机。

1.1.2 大数据

近年来，随着计算技术、数据采集、数据存储、数据传输、网络通信等信息技术以及脑科学、认知科学等相关学科领域科技的飞速发展，越来越多的联网传感器被嵌入各种设备，它们产生着数以亿计的大数据，如滚滚洪流汹涌而至。如何有效地将其接收、处理、存储乃至利用，成为摆在我们面前的一个巨大挑战。下面看一个实际的大数据案例。

如图1-2所示，用户在享受云计算、大数据带来的生活便利的同时，也在贡献着大量的数据，这些数据以文本、图像、视频等各种结构化、非结构化或半结构化的形态存在。2016年12月，中国信息通信研究院发布的《大数据白皮书(2016年)》指出“大数据是新资源、新技术和新理念的混合体”[3]。

从**资源视角**看，大数据是新资源，体现了一种全新的资源观。1990年以来，计算存储和传输数据的能力在以指数速度增长，每吉字节(GB)存储器的价格每年下降40%。2006年以来，以Hadoop为代表的分布式存储和计算技术迅猛发展，极大地提升了互联网企业数据管理能力。互联网企业对“数据废气”(data exhaust)的挖掘利用大获成功，引发了全社会开始重新审视“数据”的价值，开始把数据当作一种独特的战略资源对待。

从**技术视角**看，大数据代表了新一代数据管理与分析技术。传统的数据管理与分析技术以结构化数据为管理对象，在小数据集上进行分析，以集中式架构为主，成本高昂。与“贵族化”的数据分析技术相比，源于互联网的、面向多源异构数据的、在超大规模数据集(PB量级)上进行分析的、以分布式架构为主的新一代数据管理技术与开源软件潮流相互叠加，在大幅提高处理效率的同时，成百倍地降低了数据应用成本。

从**理念视角**看，大数据打开了一种全新的思维角度。大数据的应用赋予了“实事求是”新的内涵。其一是“数据驱动”，即经营管理决策可以自下而上地由数据来驱动，甚至像量化股票交易、实时竞价广告等场景中那样，可以由机器根据数据直接决策；其二是“数据闭环”，观察互联网行业大数据案例，它们往往能够构造起包括数据采集、建模分析、效

---

① 国务院. 国务院关于促进云计算创新发展培育信息产业新业态的意见. 2015-01-30. http://www.gov.cn/zhengce/content/2015-01/30/content_9440.htm.

② 百度百科. 云计算. http://baike.baidu.com/view/1316082.htm.

果评估、反馈修正等环节在内的完整的"数据闭环",从而能够不断地自我升级,螺旋上升。目前很多"大数据应用",要么数据量不够大,要么并非必须使用新一代技术,但体现了数据驱动和数据闭环的思维,改进了生产管理效率,这是大数据思维理念应用的体现。

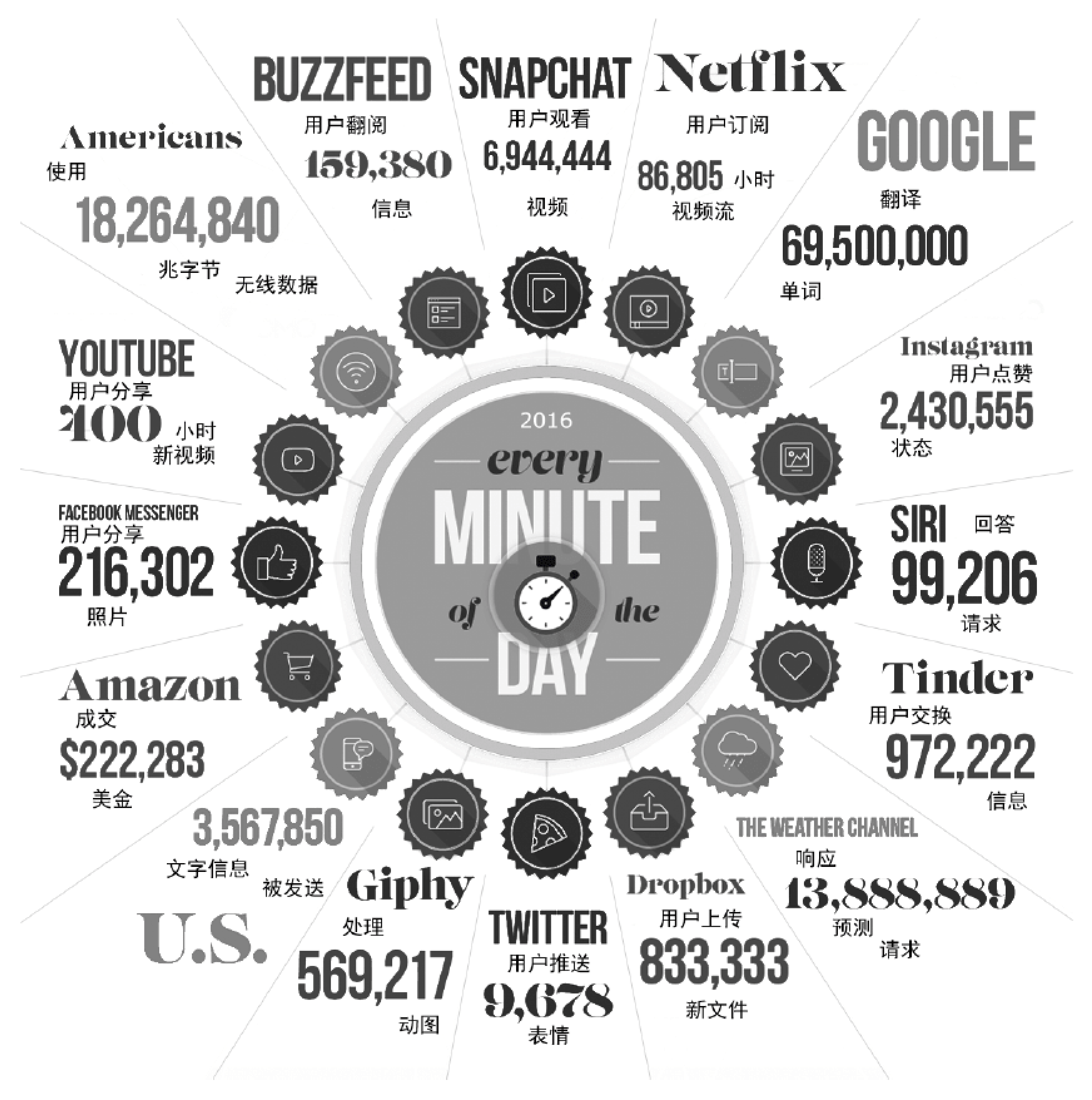


图 1-2　每分钟产生的"大"数据(摘自 Domo：Data Never Sleeps 4.0[①])

1.1.3　云计算与大数据的智能应用

2016 年初,谷歌 AlphaGo(阿发围棋,阿发狗)挑战韩国的围棋高手李世石九段,将全世界人的目光都吸引到人工智能、大数据智能处理上,这一挑战过程更是刺激了无数围棋和人工智能爱好者的神经。围棋中各种变数的数量,甚至超过了宇宙中原子的数量。作为机器,在与人对弈时,其显著的优势在于能够记录收集到的全部比赛历史并进行复盘演练。然而,人是灵活的、多变的,即使同一个人在相同的开局下也可能演变出不同的棋路。这对棋谱数据的智能处理提出了更高的要求,是人工智能和大数据智能处理技术的综合应用体现。当比赛最终尘埃落定,各方面人士和团体机构纷纷热议人工智能和大数据智

① Domo. Data Never Sleeps 4.0. 2017-01-01. https://www.domo.com/learn/data-never-sleeps-4-0.

能处理技术的发展——有悲观的，也有乐观的；有憧憬的，也有恐慌的；有支持的，也有反对的。也因此在世界范围内掀起了一轮新的人工智能、大数据智能处理热潮。2017 年 10 月，DeepMind 团队在《自然》杂志上发表的论文 *AlphaGo Zero：Mastering the game of Go without human knowledge*（《无需人类知识的围棋大师》）推出了人工智能围棋程序最新版本更强大的“学习”能力。

回顾一下计算机的发展历程，从 1936 年图灵发表第一篇关于可计算性的论文以来，各种新理论、新技术或者新应用不断涌现，从计算机的发明、软件工程的提出，到互联网、万维网的出现，再到云计算、大数据的出现，整整 80 年间计算机有了突飞猛进的发展，也推动了人类社会的变革（图 1-3）。

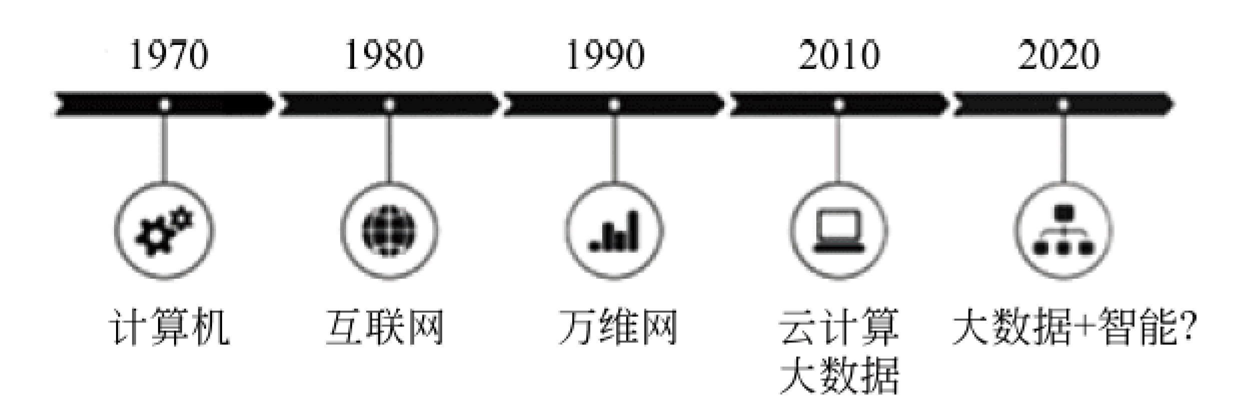


图 1-3　每个十年的技术发展代表

- 1936 年 5 月，图灵向伦敦权威的数学杂志投了一篇论文，题为《论数字计算在决断难题中的应用》。该文于 1937 年在《伦敦数学会文集》第 42 期上发表后，立即引起广泛的关注。在论文的附录里，他描述了一种可以辅助数学研究的机器，后来被人称为“图灵机”，第一次在纯数学的符号逻辑和实体世界之间建立了联系。现在我们所使用的计算机以及“人工智能”都是以此为基础的。
- 1944 年，冯·诺依曼与摩根斯特恩合著的《博弈论与经济行为》是博弈论学科的奠基性著作，对世界上第一台电子计算机 ENIAC 的设计提出过建议。1945 年 3 月，他们在共同讨论的基础上起草了一个全新的“存储程序通用电子计算机方案”——EDVAC（Electronic Discrete Variable Automatic Computer）。这对后来计算机的设计有决定性的影响，特别是确定计算机的结构，采用存储程序以及二进制编码等，至今仍为电子计算机设计者所遵循。
- 1968 年 10 月，在德国的南部小城加尔米施（Garmisch）举行了一次在软件历史上非常有名的会议，会议由北大西洋公约组织（NATO）的科技委员会出资，会议的名字就叫“软件工程大会”。软件工程在当时还是一个新鲜名词，这个会议颇有“以战略眼光审视新出现的软件危机”的意味。软件工程是一门研究用工程化方法构建和维护有效的、实用的和高质量的软件的学科。它涉及程序设计语言、数据库、软件开发工具、系统平台、标准、设计模式等方面。
- 1969 年 12 月，ARPANET 投入运行，建成了一个实验性的由 4 个节点连接的网络。到 1983 年，ARPANET 已连接了三百多台计算机，供美国各研究机构和政府部门使用。1983 年，ARPANET 分为 ARPANet 和军用的 MILNET（Military Network），两个网络之间可以进行通信和资源共享。由于这两个网络都是由许多网络互联而成的，因此它们都被称为 Internet，ARPANet 就是 Internet 的前身。

- 1989 年 3 月，伯纳斯·李撰写了《关于信息化管理的建议》一文，文中提及 ENQUIRE 并且描述了一个更加精巧的管理模型。1990 年 11 月 12 日他和罗伯特·卡里奥(Robert Cailliau)合作提出了一个关于万维网的更加正式的建议。1990 年 11 月 13 日，他在一台 NeXT 工作站上写了第一个网页以实现他文中的想法。WWW 是环球信息网的缩写(亦作 Web、WWW、W3，英文全称为 World Wide Web)，中文名字为万维网、环球网等，常简称为 Web。WWW 分为 Web 客户端和 Web 服务器程序，Web 客户端(常用浏览器)可以访问浏览 Web 服务器上的页面。
- 1998 年，科学家迎来了复杂网络的又一次突破性进展，首先冲破了 ER 随机图理论框架。美国康奈尔大学理论和应用力学系的博士生 Watts 及其导师 Strogatz 在 *Nature* 杂志上发表了题为《“小世界”网络的群体动力行为》的论文，提出了小世界网络模型，推广了“六度分离”的科学假设。“六度分离”来自对社会调查的推断，指在大多数人中，任意两个素不相识的人通过朋友的朋友，平均最多通过 6 个人就能够彼此认识。
- 2001 年 1 月，维基百科由 Bomis 网站的总裁吉米·威尔士发起。维基百科是一个基于维基技术的全球性多语言百科全书协作计划，同时也是一部用多种语言编成的网络百科全书，其目标及宗旨是为全人类提供自由的百科全书——用他们所选择的语言来书写而成的，是一个动态的、可自由访问和编辑的全球知识体。
- 2006 年 8 月，Google 首席执行官埃里克·施密特在搜索引擎大会(SES San Jose 2006)首次提出“云计算”(cloud computing)的概念。对云计算的定义有多种说法，至少可以找到 100 种解释。现阶段广为接受的是美国国家标准与技术研究院(NIST)的定义：云计算是一种按使用量付费的模式，这种模式提供可用的、便捷的、按需的网络访问，进入可配置的计算资源共享池(资源包括网络、服务器、存储、应用软件、服务)，这些资源能够被快速提供，只需投入很少的管理工作，或与服务供应商进行很少的交互。
- 2007 年 3 月，约翰·F·甘茨、大卫·莱茵泽尔及互联网数据中心(IDC)的其他研究人员出版了一本白皮书，题为《膨胀的数字宇宙：2010 年世界信息增长预测》。这是第一份评估与预测每年世界所产生与复制的数字化数据总量的研究。

传统的网络计算主要是一种基于互联网的计算系统，而人机交互的互联计算强调的是人对云计算的参与和贡献，人已经作为一种计算资源介入计算系统之中，可以参与到云计算的输入、计算处理过程和输出。这样的说法乍一看可能还感觉有点陌生，但是每个人却每时每刻都在经历着，比如：

- 2015 年春节的关键词：“摇一摇、抢红包!”。
- 春节期间微信红包收发总量为 32.7 亿次，“春晚摇一摇”互动总量超过 110 亿次(22 时 34 分春晚摇一摇互动出现峰值，达到了 8.1 亿次/分)。
- 微信红包的发放者不仅是机器，还有大量人的参与。
- 接收者如何判断是人还是机器自动发出的红包?

还有很多其他的例子，比如：

- 脑机接口。通过意念控制机器，实现了大脑参与到整个计算系统之中。
- 常识性知识表达。这项研究中存在几个关键的挑战，主要是常识性知识的数据量庞大，既无法用自然语言清晰地描述，也无法用形式化方法进行描述，它的边界更是难以确定。人机交互的互联计算的出现有望解决这一难题。

在人机交互的互联计算环境中，数据的处理不仅是机器在做，还有人的参与。世界上有机器人，也出现了人机器；世界上有计算机，现在也有了计算人。实际上，当机器被用来代替人的部分功能，就是机器人。现在已经出现了人作为整个系统的一部分参与计算和工作的情况，这就成了人机器。用来做计算的机器被称为计算机。而在人机交互的云计算系统里，人作为整个系统的一员产生贡献，此时的人就成了计算人。我们每天都在享受云计算，如通过腾讯视频观看最新的体育节目、娱乐视频、网络直播，通过微信与朋友、家人互传信息，进行视频聊天，在当当、天猫和京东等采购书籍、服装和电子产品等。云安全看似遥远，实际在人们每天在使用网络的时候，也在享受云安全服务。在现实生活的交通中，E 代驾给人们提供了很多出行方便。云计算服务人类工作生活的案例也很多，读者可以在自己的日常生活中去体会。

云计算也引发了社会的诸多进步，下面是几个影响较大的案例。

1. 案例一：维基百科

维基之父吉米·威尔士和拉里·桑格（图 1-4）在三十多岁的时候就发出豪言壮语：让世界上每个人都能自由地分享人类知识的总和。他们是怎么实现的呢？网络建立起了一个人与人可以充分沟通的公共计算环境，群体智能也融入了网络。在维基百科这个系统里面，大众既是系统的使用者，也是系统的开发者。任何用户都可以对自己感兴趣的条目进行编辑，贡献个人观点和看法。在维基百科中，尽管每个人对条目的编辑都可能会出现错误，甚至有恶意的篡改，但是在大众参与的情况下，错误和恶意篡改的部分会很快得到纠正，大部分条目保持了相当高的水准。这与人类社会的进化演化过程何其相似！人类社会的进化演化过程不会由于个体的倒退而停止前进的步伐。以云计算条目为例，该条目第一次被编辑是发生在 2007 年，至今累计有四千多个用户进行了九千多次编辑。目前，每天还有两三次的编辑。几千人为这样一个条目所做的贡献正是基于云计算平台所带来的方便而形成的。



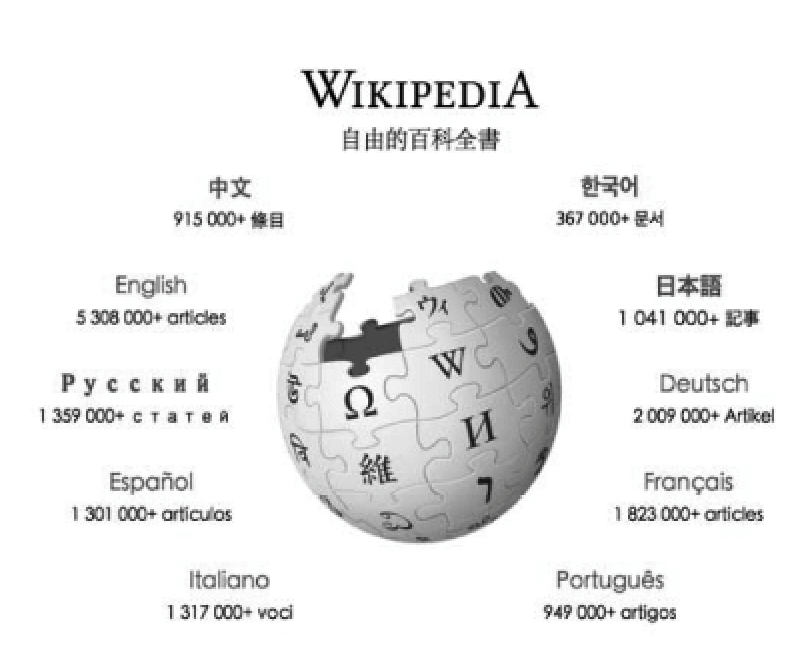




图 1-4　维基百科及其创建者（左：吉米·威尔士，右：拉里·桑格）

2. 案例二：人机交互的验证码与光学字符识别(OCR)系统

这是一个典型的跨界融合成功案例。reCAPTCHA[①] 是卡内基·梅隆大学设计的一个系统，它借助于人脑对难以识别的字符的辨别能力，对古旧书籍中难以被 OCR 系统识别的字符进行辨别。通过这个系统把古旧书籍中的字符扫描下来，发给要登录输入验证码的用户，他们用肉眼识别这些文字，然后进行比对，如果输入正确，就认为这个登录者是一个人而不是机器，这就完成了对登录者的身份识别(图 1-5)。在这个过程中，解决了两个问题：一是要判定一个用户在登录的时候是人还是机器，这是网络安全里的一个问题；二是实现了用人来对古旧书籍中的这些难以识别的字符进行识别的能力。如果需要我们组织人来做古旧书籍的文字识别，这个工程是庞大的，而现在每天都有上亿的人在全世界来帮助完成古旧书籍中的文字识别。也就是说，这个系统把这两大经典困难问题融合到了一起，用跨界思维的方式解决了这两大困难。在这样一个系统中，我们不难看出人已经成为一个 OCR 的工具。

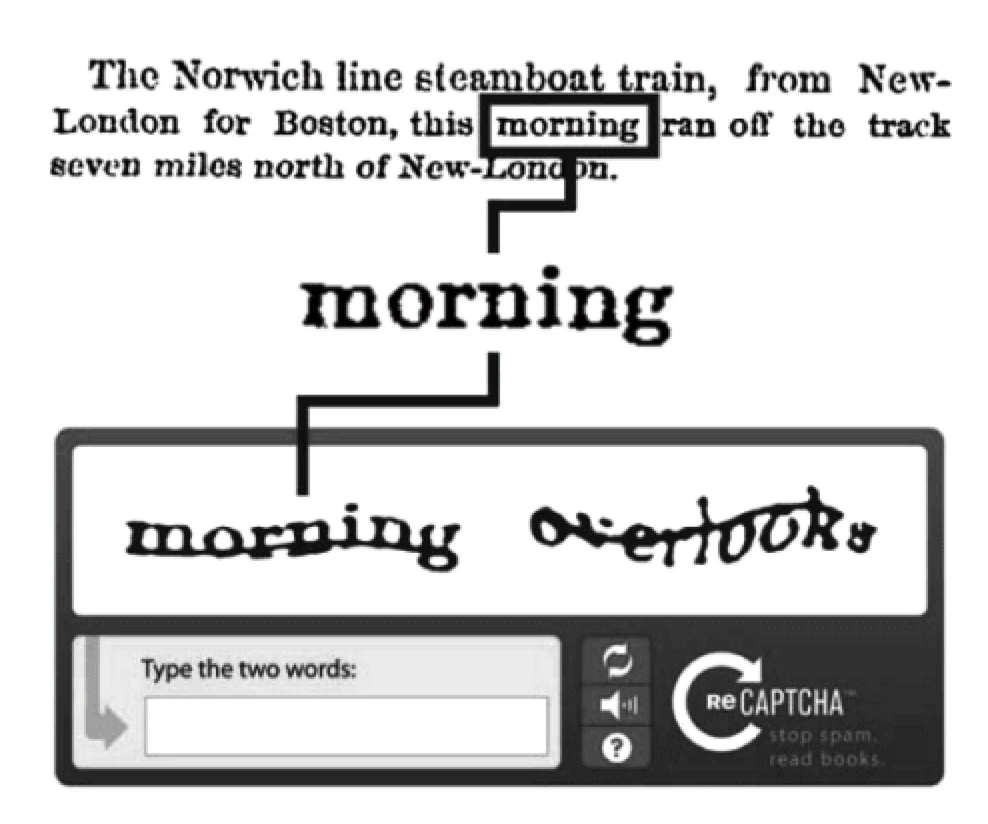


图 1-5　reCAPTCHA 系统示例

3. 案例三：慕课(MOOC)

慕课(Massive Open Online Course，大规模开放在线课堂)[②]作为一种在线课程模式，目前已经在全世界流行。如图 1-6 所示，维基百科指出，慕课海报中的每一个字母蕴含可协商性问题以探讨慕课的含义(例如 M 字母蕴含的一个可协商问题可以是“What is Massive?”等)，慕课没有完全既定的定义，但有两个显著的特点：

(1) 开放共享(Open access)。慕课参与者不必是在校的注册学生，也不要求交学费，它是让大家共享的。

(2) 可扩张性(scalability)。许多传统课堂针对一小群学生，但慕课里的“大规模”课堂是针对不确定的参与者设计的。

在慕课这样一个教育云系统上面，通过名师共享，学生可以在任何一所学校听到名

① reCAPTCHA 项目：https://www.cylab.cmu.edu/partners/success-stories/recaptcha.html

② https://en.wikipedia.org/wiki/Massive_open_online_course

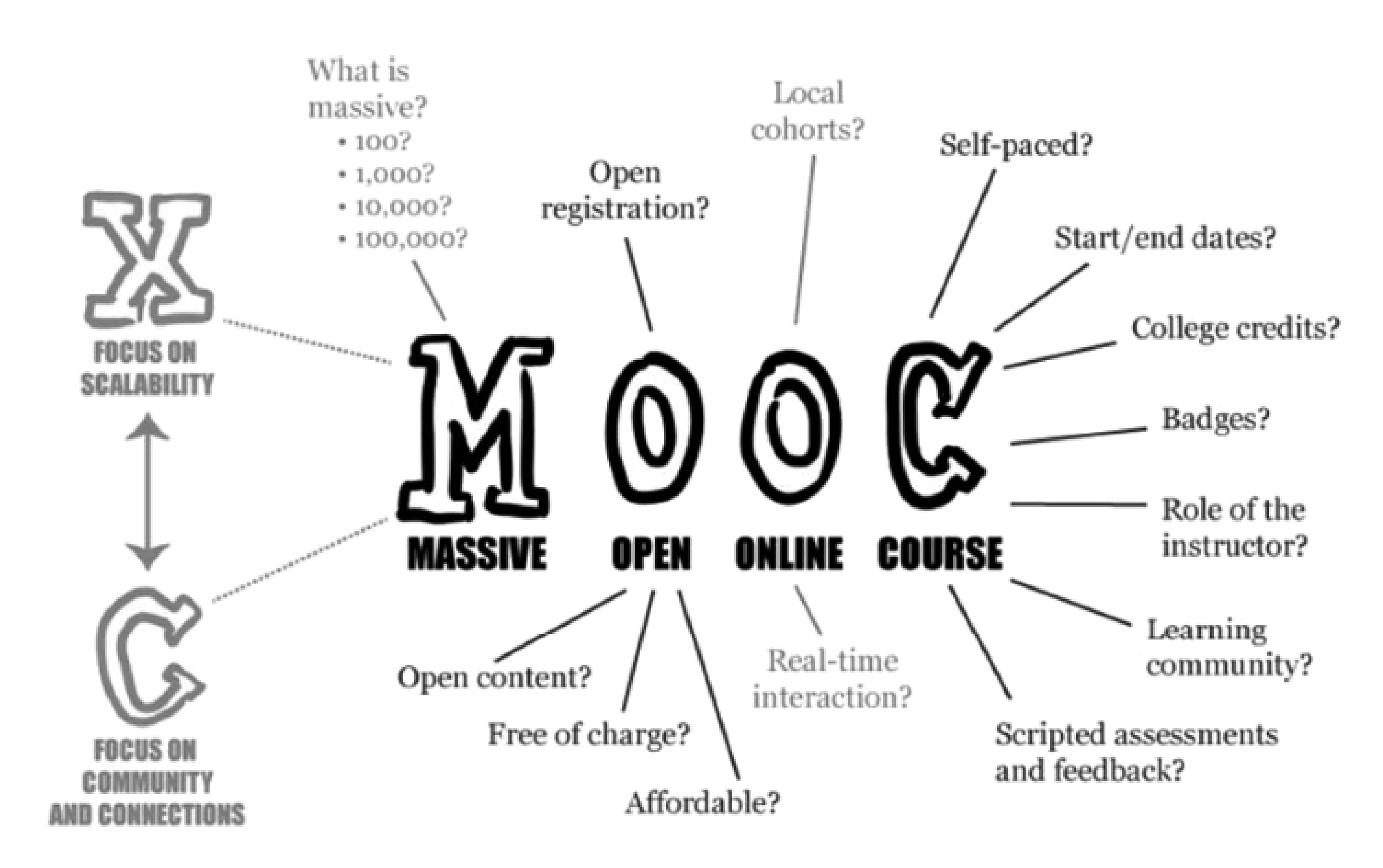


图 1-6 维基百科关于慕课每个字母的含义的定义

师的授课。同时,借助智能导学,名师可以在任意的时间辅导大家学习。随着慕课的推广,可能有人会担忧:教师会不会面临下岗?这是不可能的!实际上在整个教育系统中,学校有 3 方面的职能,即培养人才、科学研究和服务社会,这自然地就对应了传授知识、生产知识和使用知识。而慕课作为一种新型教育工具,仅对应其中的传授知识环节。

通过这些生活中的实际案例,可以发现,互联网已经突破了传统图灵机的范畴。随着"互联网+"时代的到来,数据一直在快速增长,人们意识到一些原有的信息系统解决方案针对大数据已经无效了。数据已经不能够通过几台机器或设备来处理,需要通过更多、更大规模的系统来解决大数据的问题。单一设备的计算性能已经不是大数据平台计算的瓶颈。云计算和大数据给人们带来了新的挑战。如表 1-1 所示,互联网上的云计算是以交互为中心的,而传统的、集中的调度和顺序的、确定的输入不能描述互联网的工作机理和交互机理,互联网不等同于一台虚拟的图灵机模型。人已经成为这个计算系统中的一员。

表 1-1 图灵计算与云计算的区别

图灵计算	云计算
重点关注 CPU 和操作系统	重点关注节点间的交互
确定的计算	不确定的计算
最优解	尽可能的解
统一的调度	无集中控制、局域偏好依附
机械的执行	有主体行为能力
可计算模型	服务计算模型
人不参与的计算	人参与的计算

我们来比较一下图灵计算和云计算：图灵计算关注 CPU 和操作系统，而云计算的关注重点是节点之间的交互；图灵计算是确定的计算，而云计算是不确定的计算；图灵计算寻找的是最优解，而云计算寻找的是尽可能的解；图灵计算是统一的调度，而云计算的调度是没有集中控制的、局域偏好依附的；图灵计算是机械的执行，而云计算具有主体行为能力；图灵计算采用的是可计算模型，而云计算采用的是服务计算模型；最后，也是最关键的一点，图灵计算是人不参与的计算，而云计算是人参与其中的计算。

1.2 大数据的发展及挑战

本节介绍大数据的发展历史及现状，并从系统平台、分析处理两个方面提出大数据挖掘面临的挑战。通过本节的学习，读者可以深入理解大数据的特征、发展状况和面临的挑战。

1.2.1 大数据的发展催生三元空间世界

2012 年 3 月 29 日，美国政府公布了大数据研发计划。该计划旨在改进人们从海量和复杂的数据中获取知识的能力，进而加速美国在科学与工程领域发明的步伐，强化国家安全，转变现有的教学和学习方式。近年来，随着互联网进入 Web 3.0、制造业进入工业 4.0 时代，以及物联网和云计算的迅猛发展，人类社会逐步进入了大数据时代。维基百科指出，大数据（big data），或称巨量数据、海量数据、大资料，指的是所涉及的数据量规模巨大到无法通过人工或者计算机在合理的时间内达到截取、管理、处理并整理成为人类所能解读的形式的信息[6]。业界对大数据的认识也在不断地变化并完善。2001 年，麦塔集团（META Group）的数据分析师莱尼（Doug Laney）首先在一份报告中提出“3-D 数据管理”的大数据观点，认为数据成长将朝 3 个方向发展，分别是数据即时处理的速度（Velocity）、数据格式的多样化（Variety）和数据量的规模（Volume），这被认为是大数据 3V 特点的来源。近年来，随着科学技术的发展，获取、分析、存储数据的手段发生了巨大的变化，3V 已经不足以形容现今的大数据。在 2012 年，包括科技巨头 IBM、Google 和国际调查机构 Gartner、IDC 等纷纷对大数据提出新的论述，将 3V 提升至 4V 甚至 5V(图 1-7)。无论 3V、4V 还是 5V，一般地，人们认为大数据具备以下特征：

- 稠密与稀疏共存——局部稠密与全局稀疏。
- 冗余与缺失并在——大量冗余与局部缺失。
- 显式与隐式均有——大量显式与丰富隐式。
- 静态与动态互现——动态演进与静态关联。
- 多元与异质共处——多元多变与异质异性。
- 量大与可用矛盾——量大低值与可用稀少。

从数据本身的角度而言，大数据技术能够发现数据之间存在的直接或间接的关联性，通过采用一系列技术和方法挖掘并展现数据中蕴含的价值，包括数据采集、预处理、存储、分析挖掘、可视化等。这是一个通过对特定的大数据集或应用进行分析，获得有价值信息的过程。大数据技术的研究与突破，其最终目标即是从复杂的数据集中发现新的模式与

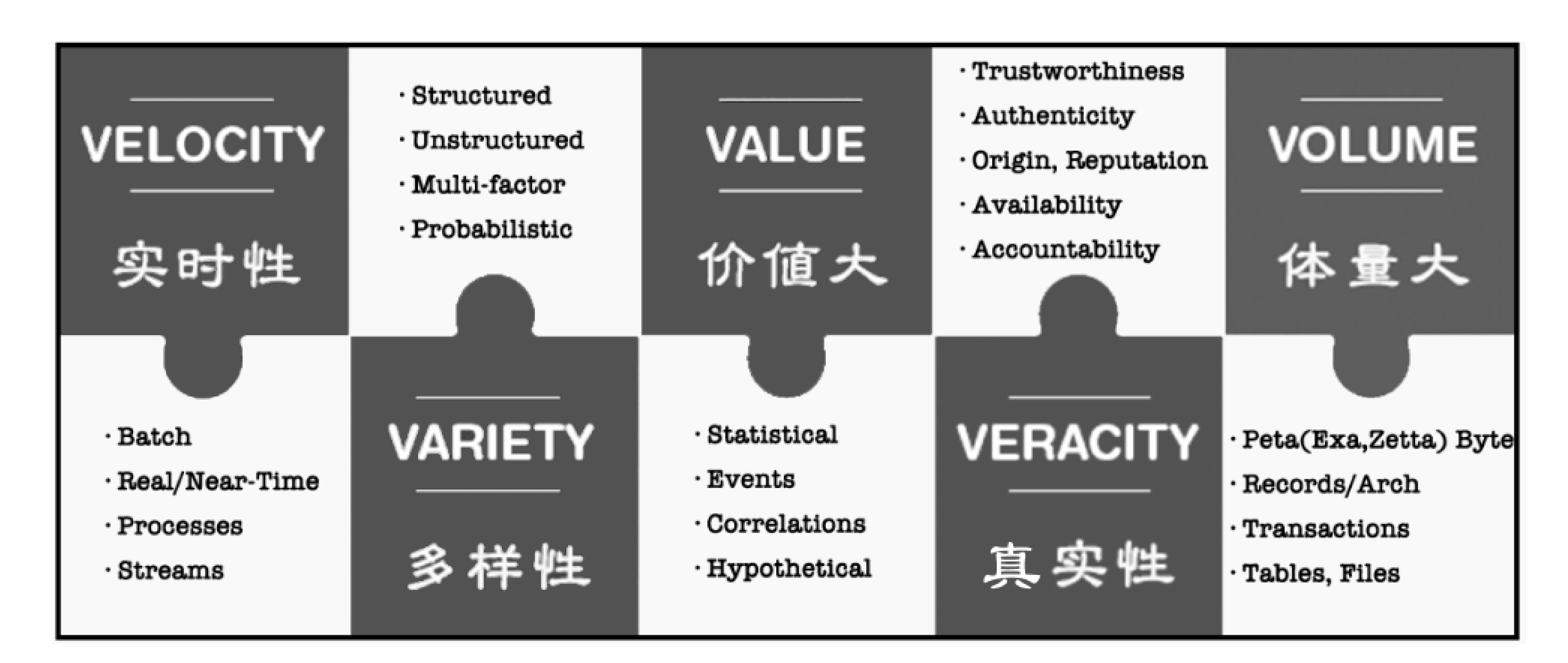


图 1-7　大数据的 5V 特点

知识，挖掘得到有价值的新信息[2]。对于大数据的定义，不同的人有不同的看法，但是有一个共识：大数据需要新的处理模式，才能具有更强的决策力、洞察发现力和流程优化能力来适应海量、高增长率和多样性的信息资产。人们经常提到的 3V、4V、5V，甚至以后可能更多个 V，核心都是表明大数据质量要求高，具体表现在数据量巨大、种类繁多、变化速度快、价值密度高、同时准确性要求高等方面。

互联网、云计算的发展催生了大数据的出现。但是事实上大数据已经出现了千万年，这得益于人类的眼睛、耳朵等最原始、最自然的生理器官这些高精度、智能的视听传感器。遗憾的是，以前人类虽然有这些高精度传感器，但是这些传感器的数据没有进行数字化，没有互联，也就没有大数据的问题。现在互联网把这些数据连起来了，也就有了大数据的问题。各个行业里也存在大数据，比如金融、医疗、保险、交通、气象、制造等各行各业有各种数据，当这些数据没有连起来时就是信息的孤岛，一旦连起来，就是人们可以利用的大数据。人们对物理空间的认识导致了自然科学的出现；人们对社会的认知导致了社会科学的诞生；在今天的数据空间，有了很好的数据，这些数据甚至已经互联，它们相互之间能够有相互的作用了，这时候也会导致数据和计算科学的诞生。在数据空间中，甚至出现了一个新的人类，也就是网民。

对于悲观者而言，大数据意味着数据存储世界的末日；对乐观者而言，这里孕育了巨大的市场机会，庞大的数据就是一个信息金矿。随着技术的进步，其财富价值将很快被人们发现，而且越来越容易。本书认为大数据是需要新的处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。简而言之，大数据分析的过程将越来越智能，大数据分析的结果将越来越多地服务于人们的社会和生活。驱动社会物理学的引擎是大数据——近年来无所不在的、关于人类生活的各个方面的数字数据。社会物理学的作用体现在分析人类活动的规律以及人类活动所留下的数字“面包屑”(通话记录、通勤记录、网购记录、刷卡记录等)里包含的想法中[4]。我们究竟是怎么样的一个人，不是通过我们声称自己做了什么，而是通过我们去过的地点、购买的物品、发送的短信息等更为准确地决定的，也就是我们的精准个人画像。

世界的演化发展经历了一元空间、二元空间，正在向三元空间发展。人类社会诞生之前，世上仅有物理空间（一元）；随着人类社会的形成和发展，产生了社会空间（二元）；人类社会进入信息社会，正逐步形成数据空间（三元），数据成为物质与能源之外的新型资源。人类对传统二元空间的认识形成了自然科学、社会科学，对数据空间的认识将逐渐形成数据科学/计算科学。在数据空间演化成熟并被人类深入认识之前，实践和技术的发展正在倒逼科学的发展，正如现代科学技术的诞生与形成过程一样。数据/计算科学在未来也将反过来推动人类文明的进步，正如现代航空航天、核能技术、电子技术等诸多现代科技对社会发展的推动一样。然而，目前人们还有没很好地充分利用身边的数据信息。人们花了很大的代价来获取、存储、传输、处理各种数据，但是还没有能够很好地实现它的价值。在建设节约型社会的今天，物质、能量、环境资源不能够浪费，同样，数据资源也不能浪费。

1.2.2 大数据智能分析处理面临的挑战

大数据在带来发展机遇的同时，也带来了新的挑战，催生了新技术的发展和旧技术的革新。不断增长的数据规模和数据的动态快速产生要求必须采用分布式计算框架才能实现与之相匹配的吞吐和实时性，而数据的持久化保存也离不开分布式存储。庞大的数据规模与数据质量的参差不齐对模型提出了新的要求，不再是一成不变、一步到位的高精度计算模型，而是可以更加智能，能够随着数据的到来和变化进行一定的自主学习和训练的智能计算模型。这对传统的理论模型、计算方法提出了新的挑战[10][11][12][13]，主要体现在系统平台和分析处理两个方面。

1. 系统平台方面

大数据智能计算给系统平台带来以下几个挑战。

（1）**大数据处理与硬件协同**。因数据中心硬件架构、厂商、采购年代不同带来的硬件易购性，将使木桶效应在集群中变得极为复杂；技术变革促进了新硬件的产生，数据中心的基础硬件体系结构已经从传统的 CPU-内存-HDD 硬盘 3 级结构逐渐升级为 CPU＋GPU-内存-SSD 硬盘-HDD 硬盘的混合计算、混合存储体系结构。

（2）**大数据集成**。数据类型的多样性和数据采集设备的多样性，使得数据集成过程中的数据转换变得非常复杂和难以管理；数据质量因数据规模的增加反而变得越来越差，亟须研究更便捷、有效的方式实现数据清洗，实现质与量的平衡。

（3）**大数据隐私**。随着数据规模的增加和新的数据挖掘模型的产生，如何有效地保护数据中的用户隐私，并应对数据动态变化带来的隐私保护模型失效，将更具挑战。

（4）**大数据能耗**。大数据中心除了正常提供服务的能耗外，还提供了一定的闲置比例以应对突如其来的计算、网络流量高峰。高能耗已经成为制约大数据快速发展的一个主要瓶颈，新型低功耗硬件的研发和再生能源的使用正在引起人们的重视。

（5）**大数据管理**。一方面，在大数据时代，数据处理、挖掘及其结果形式更加多样化。复杂的分析过程和难以理解的分析结果限制了人们从大数据中快速获取知识的

能力。从数据集成到数据分析,直到最后的数据解释,易用性贯穿整个大数据管理过程。可视化、人机交互及数据起源技术都可以有效地提升易用性,而这3类技术的背后又离不开海量元数据管理技术的支撑。另一方面,关系数据库产品的成功离不开以TPC系列为代表的测试基准的产生。目前尚缺少全面的大数据管理的测试基准,其面临的主要挑战包括系统复杂度高、用户案例多样、数据规模庞大、系统快速演变、测试基准的重构与复用等。

2. 分析处理方面

计算机中存在不断变大的数据集,不存在绝对的大数据,计算机中的所有数据集都是有限集合。小数据集合上的处理和分析方法在直接移植到大数据上时因数据本身的特点而被放大,这个问题将对传统的方法带来极大的挑战,甚至使传统的方法失效。

(1) **大数据质量**。数据量大不一定就代表信息量或者数据价值的增大,相反,很多时候意味着信息垃圾的泛滥。数据清洗在大数据处理过程中对计算性能和计算精度起着至关重要的作用。如果数据清洗的粒度过细,很容易将有用的信息过滤掉;而如果清洗粒度过粗,又无法达到真正的清洗效果。因此,在质与量之间需要进行仔细的考量和权衡。

(2) **大数据实时性**。时间流逝带来了数据蕴含的知识的衰减。为了解决这一问题,诸多应用场景从离线向在线转变,这对大数据分析的实时性提出了较高的要求。结构化数据一般都有较好的先验知识和较固定的索引方法,而半结构化和非结构化数据需要额外的时间去建立先验知识和索引方法,给数据处理和分析的时效性带来了巨大的挑战。实时处理的模式选择主要有3种思路:流处理模式、批处理模式(准实时)和二者的融合。

(3) **大数据采样**。研究如何把大数据变小,找到与算法相适应的极小样本集,并降低采样对算法误差的影响。

(4) **大数据不一致性**。数据质量的一个重要方面是不一致性检测。使用不一致的数据可能导致错误的决策,甚至会付出昂贵的代价,并最终导致算法失效或无解。

(5) **大数据超高维性**。大数据处理中的超高维问题突出,因超高维导致的数据稀疏问题增加了急剧计算和分析方法的复杂度。

(6) **大数据不确定性**。大数据处理中经常发生原始数据不准确、粗粒度数据集合向细粒度集合的转变、数据值的缺失、多模态数据集成和模态转化等问题,亟须研究新的数据管理和数据分析模型来应对大规模数据的不确定性问题。

继几千年前的经验科学、数百年前的理论科学和数十年前的计算科学之后,当今的数据爆炸孕育了数据密集型科学,将理论、实验和计算仿真等范式统一起来(表1-2)。大数据具有“取之不尽,用之不竭”的特性,在不断的再利用、重组和扩展中持续释放其潜在价值,在广泛的公开、共享中不断创造着新的财富,被誉为“非竞争性”生产要素。大数据被定义为科学研究的第四范式,即eScience。

表 1-2　科学研究的范式

科学范式	年代	方法	用　　途
第一范式	数千年前，亚里士多德时代	基于经验的	用于描述自然现象
第二范式	数百年前，牛顿时代	基于理论研究的	着眼于建立数学模型并进行推广
第三范式	几十年前	基于计算的	借助强大的计算能力，可以模拟复杂的自然现象
第四范式（eScience）	当今	基于数据探索的	利用仪器获取数据或者利用模拟器生成数据，再利用软件进行处理，将知识或信息存储在计算机中，科学家利用数据管理技术和统计方法进行科学探索

1.3　数据挖掘概述

本节对传统数据挖掘和大数据挖掘进行概述，为读者更好地认识大数据挖掘奠定基础。1.3.1 节介绍数据挖掘的基本概念，1.3.2 节介绍数据挖掘的功能，1.3.3 节介绍数据挖掘运用的技术，最后对大数据挖掘与传统数据挖掘方法进行比较。

1.3.1　数据挖掘的概念

随着信息科技的进步和网络的发达、计算机运算能力的增强以及数据存储技术的不断改进，人类社会正迈向信息时代。数据的爆炸式增长、广泛利用和巨大体量使得我们的时代成为真正的数据时代，迫切需要功能强大和通用的工具，以便从大数据中发现有价值的信息，将这些数据转换成有用的信息和知识，所获取的信息和知识可以广泛用于各种应用，包括商务管理、生产控制、市场分析、工程设计和科学探索等。数据挖掘方法利用了来自许多领域的技术思想，如来自统计学的抽样、估计和假设检验，来自人工智能、模式识别和机器学习的搜索算法、建模技术和学习理论，来自包括最优化、进化计算、信息论、信号处理、可视化和信息检索等的重要支撑。随着数据量的越来越大，源于高性能分布式并行计算和存储的技术在大数据挖掘和应用中显得尤为重要。

许多人把数据挖掘视为另一个流行术语——**数据中的知识发现**（KDD）的同义词，而另一些人只是把数据挖掘视为知识发现过程的一个基本步骤。一般认为，知识发现由以下步骤的迭代序列组成：

（1）数据清理——消除噪声和删除不一致数据。

（2）数据集成——多种数据源可以组合在一起，形成数据集市或数据仓库。

（3）数据选择——从数据库中提取与分析任务相关的数据。

（4）数据变换——通过汇总或聚集操作，把数据经过变换统一成适合挖掘的形式。

（5）数据挖掘——使用智能方法提取数据模式。

（6）模式评估——根据某种兴趣度量，识别代表知识的真正有趣的模式。

（7）知识表示——使用可视化和知识表示技术向用户提供挖掘的知识。

1.3.2 数据挖掘的功能

数据挖掘功能用于指定数据挖掘任务发现的模式。一般而言，这些任务可以分为描述性的和预测性的。描述性挖掘任务刻画目标数据中数据的一般性质；预测性挖掘任务在当前数据上进行归纳，以便做出预测。如图 1-8 所示，常见的数据挖掘功能包括聚类、分类、关联分析、数据总结、偏差检测和预测等。其中聚类、关联分析、数据总结、偏差检测可以认为是描述性任务，分类和预测可以认为是预测性任务。

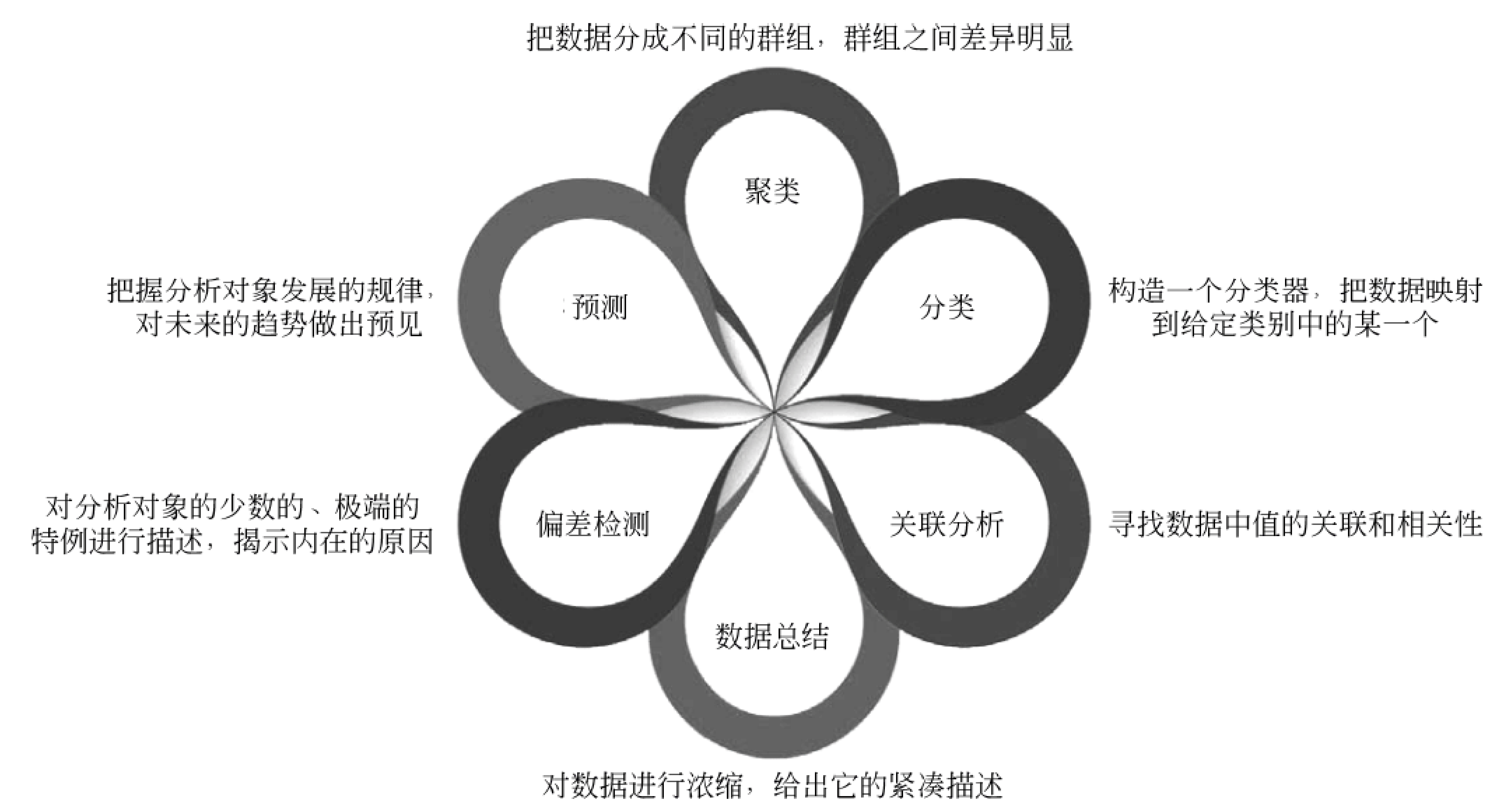


图 1-8 数据挖掘的主要功能分类

- **聚类**。聚类是一个把数据对象（或观测）划分成子集的过程，每个子集是一个簇。数据对象根据最大化类内相似性、最小化类间相似性的原则进行聚类或分组。因为没有提供类标号信息，通过观察学习而不是通过示例学习，聚类是一种无监督学习。
- **分类**。分类是一种重要的数据分析形式，它提取刻画重要数据类的模型。这种模型称为分类器，预测分类的（离散的、无序的）类标号，是一种监督学习，即分类器的学习是在被告知每个训练元组属于哪个类的“监督”下进行的。
- **关联分析**。若两个或多个变量的取值之间存在某种规律性，就称为关联。关联可分为简单关联、时序关联、因果关联等。关联分析的目的是找出数据中隐藏的关联网。有时并不知道数据的关联函数，即使知道也是不确定的，因此关联分析生成的规则带有可信度。
- **数据总结**。从数据分析中的统计分析演变而来，其目的是对数据进行浓缩，给出它的紧凑描述。其中，数据描述就是对某类对象的内涵进行描述，并概括这类对象的有关特征。数据描述分为特征性描述和区别性描述，前者描述某类对象的共同特征，后者描述不同类对象之间的区别。

- **偏差检测**。偏差包括很多潜在的知识，如分类中的反常实例、不满足规则的特例、观测结果与模型预测值的偏差、量值随时间的变化等。偏差检测的基本方法是，寻找观测结果与参照值之间有意义的差别，对分析对象中少数的、极端的特例进行描述，解释内在原因。
- **预测**。通过对样本数据(历史数据)的输入值和输出值的关联性学习，得到预测模型，再利用该模型对未来的输入值进行输出值预测。

1.3.3 数据挖掘运用的技术

数据挖掘研究与开发的边缘学科特性极大地促进了数据挖掘的成功和广泛应用。近年来，数据挖掘吸纳了统计学、机器学习、模式识别、数据库和数据仓库、信息检索、可视化、算法分析、高性能计算等许多领域的大量技术。

- **统计学**。研究数据的收集、分析、解释和标示。统计学方法可以用来汇总或描述数据集，也可以用来验证数据挖掘结果。推理统计学用某种方式对数据建模，解释观测中的随机性和确定性，并用来提取关于所观察的过程或总体的结论。统计假设检验使用实验数据进行统计判决，如果结果不大可能随机出现，则称它为统计显著的。
- **机器学习**。考察计算机如何基于数据学习。研究热点领域之一是基于数据自动地学习识别复杂的模式，并做出智能的决断。与数据挖掘高度相关的、经典的机器学习问题包括监督学习、无监督学习、半监督学习、主动学习等。
- **数据库与数据仓库**。许多数据挖掘任务都需要处理大型数据集，甚至是处理实时的快速流数据。因此，数据挖掘可以很好地利用可伸缩的数据库技术，以便获得在大型数据集上的高效率和可伸缩性。数据仓库集成来自多种数据源和各个时间段的数据，在多维空间合并数据，形成部分物化的数据立方体。多维数据挖掘以 OLAP 风格在多维空间进行数据挖掘，允许在各种粒度进行多维组合探查，更有可能发现代表知识的有趣模式。
- **信息检索**。是指搜索文档或文档中信息的技术，其中文档可以是结构化文本数据或非结构化多媒体数据，并且可能驻留在 Web 上。通过集成信息检索模型和数据挖掘技术，可以找出文档集中的主要主题，对集合中的每个文档，找出所涉及的主要主题等。
- **可视化**。数据的采集、提取和理解是人类感知和认识世界的基本途径之一，数据可视化为人类洞察数据的内涵、理解数据蕴藏的规律提供了重要的手段[10]。现有的数据挖掘技术在应对海量、高维、多源和动态数据的分析时，需要综合可视化、图形学、大数据挖掘理论与方法，借助新的理论模型、可视化方法和交互手段，辅助用户从大尺度、复杂、矛盾甚至不完整的数据中快速挖掘有用的信息，以便做出有效决策。也因此诞生了一门新兴学科：可视分析学。

1.3.4 大数据挖掘与传统数据挖掘

大数据带来了三大根本改变：第一，大数据让人们脱离了对算法和模型的依赖，数据

本身即可帮助人们贴近事情的真相;第二,大数据弱化了因果关系。大数据分析可以挖掘出不同要素之间的相关关系,人们不需要知道这些要素为什么相关,就可以利用其结果,在信息复杂错综的现代社会,这样的应用将大大提高效率;第三,与之前的数据库相关技术相比,大数据可以处理半结构化或非结构化的数据,这将使计算机能够分析的数据范围迅速扩大。大数据挖掘与传统数据挖掘的主要区别体现在以下几方面:

- **大数据挖掘在一定程度上降低了对传统数据挖掘模型以及算法的依赖**。人们如果想要得到精准的结论,需要建立模型来描述问题,同时,需要理顺逻辑、理解因果并设计精妙的算法来得出接近现实的结论。然而,大数据的出现在一定程度上改变了人们对于建模和算法的依赖。当数据越来越大时,数据本身(而不是研究数据所使用的算法和模型)保证了数据分析结果的有效性。即便缺乏精准的算法,只要拥有足够多的数据,也能得到接近事实的结论,数据因此而被誉为新的生产力。
- **大数据挖掘在一定程度上降低了因果关系对传统数据挖掘结果精度的影响**。例如,Google在帮助用户翻译时,并不是设定各种语法和翻译规则,而是利用Google数据库中收集的所有用户的用词习惯进行比较推荐。Google检查所有用户的写作习惯,将最常用、出现频率最高的翻译方式推荐给用户。在这一过程中,计算机可以并不了解问题的逻辑,但是当用户行为的记录数据越来越多时,计算机就可以在不了解问题逻辑的情况之下提供最为可靠的结果。可见,海量数据和处理这些数据的分析工具为理解世界提供了一条完整的新途径。
- **大数据挖掘能够在最大程度上利用互联网上记录的用户行为数据进行分析**。大数据出现之前,计算机所能够处理的数据都需要在前期进行结构化处理,并记录在相应的数据库中。但大数据技术对于数据结构化的要求大大降低,互联网上人们留下的社交信息、地理位置信息、行为习惯信息、偏好信息等各种维度的信息都可以实时处理,从而立体完整地勾勒出每一个个体的各种特征。

1.4 大数据挖掘的计算框架

本节介绍大数据挖掘的计算框架和基本流程,为读者学习大数据挖掘提供理论基础和计算框架选择依据。1.4.1节介绍大数据挖掘计算框架的主流架构和核心组件,1.4.2节介绍大数据挖掘处理的基本流程和工具。

1.4.1 大数据挖掘计算框架

大数据计算技术采用分布式计算框架来完成大数据的处理和分析任务。作为分布式计算框架,不仅要提供高效的计算模型、简单的编程接口,还要考虑可扩展性和容错能力。作为大数据处理的框架,需要有高效、可靠的输入输出(I/O),满足数据实时处理的需求,如图1-9所示。

目前,在大数据处理领域形成了以Hadoop、Spark等为代表的大数据生态圈。Hadoop是一个由Apache基金会所开发的分布式系统基础架构(图1-10)。用户可以在



图 1-9　大数据处理关键架构

不了解分布式底层细节的情况下，开发分布式程序，充分利用集群的威力进行高速运算和存储。Hadoop 的框架最核心的设计就是 HDFS 和 MapReduce。HDFS 为海量的数据提供了存储，而 MapReduce 为海量的数据提供了计算。Hadoop 正式诞生于 2006 年 1 月 28 日，是多个开源项目的生态系统，从根本上改变了企业存储、处理和分析数据的方式。与传统系统的区别是：Hadoop 可以在相同的数据上同时运行不同类型的分析工作。

Hadoop 大数据生态圈(或者泛生态圈)基本上是为了处理超过单机尺度的数据处理而诞生的。可以把它比作一个厨房工具生态圈，做饭所需要的各种锅碗瓢盆等工具各有



图 1-10 Hadoop 生态现状

各的用处,互相之间还有重合。当然,可以用汤锅直接当碗吃饭喝汤,也可以用小刀或者刨子去皮。但是每个工具又有自己的特性,虽然奇怪的组合也能工作,但是未必是最佳选择。表 1-3 列举了目前主流的大数据工具或组件。

表 1-3 大数据核心生态组件概览表

组　件	功　能
Apache ZooKeeper	分布式、开源的协调服务。主要用来解决多个分布式应用遇到的互斥协作与通信问题,大大简化分布式应用协调及其管理的难度
Apache HBase	分布式存储系统(列数据库)。高可靠性,高性能,面向列,可伸缩。可在廉价 PC Server 上搭建大规模结构化存储集群
Apache Pig	基于 Hadoop 的大规模数据分析工具。提供类 SQL 类型语言,该语言的编译器会把用户写好的 Pig 型类 SQL 脚本转换为一系列经过优化的 MR 操作并负责向集群提交任务
Apache Hive	基于 Hadoop 的一个数据仓库工具。将结构化的数据文件映射为一张数据库表,通过类 SQL 语句快速实现简单的 MR 统计,适合数据仓库的统计分析
Apache Oozie	工作流引擎服务。用于管理和协调运行在 Hadoop 平台上各种类型任务(HDFS、Pig、MR、Shell、Java 等)
Apache Flume	分布式日志数据聚合与传输工具。可用于日志数据收集、处理和传输,功能类似于 Chukwa,但比 Chukwa 更小巧实用
Apache Mahout	基于 Hadoop 的分布式程序库。提供了大量机器学习算法的 MR 实现,并提供了一系列工具,简化了从建模到测试的流程
Apache Sqoop	数据相互转移的工具。将一个关系型数据库(MySQL、Oracle、Postgres 等)中的数据导入 Hadoop 的 HDFS 中,也可以将 HDFS 的数据导入关系型数据库中
Apache Cassandra	一套开源分布式 NoSQL 数据库系统。用于存储简单格式数据,集 Google BigTable 的数据模型与 Amazon Dynamo 的完全分布式的架构于一身
Apache Avro	数据序列化系统。用于大批量数据实时动态交换,它是新的数据序列化与传输工具,可能会逐步取代 Hadoop 原有的 RPC 机制
Apache Ambari	Hadoop 及其组件的 Web 工具。提供 Hadoop 集群的部署、管理和监控等功能,为运维人员管理 Hadoop 集群提供了强大的 Web 界面
Apache Chukwa	分布式的数据收集与传输系统。可以将各种各样类型的数据收集并导入 Hadoop

续表

组　件	功　能
Apache Hama	基于 HDFS 的 BSP 并行计算框架。可用于包括图、矩阵和网络算法在内的大规模、大数据计算
Apache Giraph	基于 Hadoop 的分布式迭代图处理系统。灵感来自 BSP（Bulk Synchronous Parallel）和 Google 的 Pregel
Apache Crunch	基于 Google 的 FlumeJava 库编写的 Java 库。用于创建 MR 程序，与 Hive、Pig 类似，Crunch 提供了用于实现如连接数据、执行聚合和排序记录等常见任务的模式库
Apache Whirr	一套运行于云服务的类库。提供高度的互补性，Whirr 支持 Amazon EC2 和 Rackspace 服务
Apache Bigtop	针对 Hadoop 及其周边组件的打包、分发和测试工具。解决组件间版本依赖、冲突问题，实际上当用户用 rpm 或 yum 方式部署时，脚本内部会用到它
Apache HCatalog	基于 Hadoop 的数据表和存储管理工具。可用于管理 HDFS 元数据，它跨越 Hadoop 和 RDBMS，可以利用 Pig 和 Hive 提供关系视图
Cloudera Hue	Hadoop 及其生态圈组件的 Web 编辑工具。实现对 HDFS、Yarn、MapReduce、Hbase、Hive、Pig 等的 Web 化操作
Apache Storm	大数据实时计算平台。一套专门用于事件流处理的分布式计算框架，有很多适用场景：实时分析、在线机器学习、连续计算、分布式 RPC、分布式 ETL、易扩展、支持容错
Apache Spark	大数据内存计算平台。一个基于内存的开源计算框架，克服了 MR 模型的缺陷，能在多场景处理大规模数据，基于内存的抽象数据类型 RDD 处理速度是 Hadoop 的几十倍

近年来最火的大数据平台 Spark 2009 年诞生于伯克利大学 AMPLab，最初属于伯克利大学的研究性项目。它于 2010 年正式开源，并于 2013 年成为了 Apache 基金项目，2014 年成为 Apache 基金的顶级项目，整个过程不到五年时间。Spark 提供的基于 RDD 的一体化解决方案，将 MapReduce、Streaming、SQL、Machine Learning、Graph Processing 等模型统一到一个平台上，以一致的 API 公开，并提供相同的部署方案，使得 Spark 的工程应用领域变得更加广泛。从 2013 年 6 月到 2014 年 6 月，参与贡献的开发人员从原来的 68 位增长到 255 位，参与贡献的公司也从 17 家上升到 50 家，包括来自中国的阿里、百度、网易、腾讯、搜狐等知名互联网公司(图 1-11)。自 2013 年起，Spark 峰会每年聚集全球数千名科学家、研发工程师、数据分析师等共同探讨、分享最新的研究成果和工业应用。

“工欲善其事，必先利其器”，从大数据生态圈再回到我们的厨房生态圈。为了做不同的菜——中国菜，日本菜，法国菜，你需要各种不同的工具。而且，客人的需求正在复杂化，即使厨具不断被发明，也没有一个万用的厨具可以处理所有情况，因此厨具会变得越来越复杂。因此，在面对任何一个大数据应用场景时，你首先想到的不应该是用什么工具，而是用什么框架。如表 1-4 所示，在对应用场景和需求均明确的情况下，才能选择整体最优的系统框架和解决方案。

Users

1000+ companies

Distributors + Apps

50+ companies

图 1-11 Spark 全球影响力

表 1-4 典型大数据计算框架对比

计算框架	计算效率（实时性）	容错性	特　点	适用场景
MapReduce	低	任务出错重做	编程接口简单，计算模型受限	文本处理、Log 分析、机器学习
Spark	高	RDD 的 Lineage 保证	内存计算通用性好，更适合迭代式任务	迭代式离线分析任务、机器学习
Dryad	较高	任务出错重做	针对 Join 进行了优化，允许动态优化调度逻辑（修改 DAG 拓扑）	机器学习、微软技术栈
GraphLab	较高	检查点技术	机器学习图计算专用框架	机器学习、大图计算
Storm	高	Worker 重启或分配到新机器，任务重做	通用性好，消息传递可靠，支持热部署，主节点可靠性差	通用的实时数据分析处理
S4	高	部分容错，检查点技术	通用性较好，通信在 TCP 和 UDP 之间权衡，持久化方式简单	实时广告推荐、容忍数据丢失
Samza	高	任务出错重做	可扩展性好，兼容流处理和批处理	在线和离线任务相结合的场景
Spark Streaming	低	RDD 和预写日志（Write Ahead Logs）	通用性好，容错性好，通过设置短时间片实现实时，应用较为局限	历史数据和实时数据相结合的分析

1.4.2 大数据挖掘处理基本流程

大数据时代，数据的处理与传统的处理方式有着显著的不同：更注重全体数据的处理而非抽样数据、更注重处理的效率而非绝对精度。一个通用的大数据处理流程，可以概

括为以下几个步骤。

1. 数据采集

大数据的采集是指接收来自客户端(Web、App或者传感器形式等)的数据,并且用户可以对这些数据进行简单的查询和处理工作。在大数据的采集过程中,其主要特点和挑战是并发数高,因为同时可能会有成千上万的用户来进行访问和操作,比如每年春运期间的12306火车票售票网站和“双11”期间的天猫商城,它们并发的访问量在峰值时达到上百万甚至更高,所以需要在采集端部署大量数据库才能支撑。代表工具包括Flume、Kafka等。

2. 数据存储

互联网的数据“大”是不争的事实。目前除了互联网企业外,数据处理领域还是传统关系型数据库管理系统(RDBMS)的天下。随着互联网的出现和快速发展,尤其是移动互联网的发展,加上数码设备的大规模使用,今天数据的主要来源已经不是人机会话了,而是通过设备、服务器、应用自动产生的。传统行业的数据同时也多起来了,这些数据以非结构、半结构化为主,而真正的交易数据量并不大,增长并不快。机器产生的数据正在以几何级数增长,比如基因数据、各种用户行为数据、定位数据,图片、视频、气象、地震、医疗数据等。近年来,通过扩展和封装Hadoop来实现对互联网大数据存储、分析的技术越来越成熟。对于非结构、半结构化数据处理、复杂的ETL流程、复杂的数据挖掘和计算模型大数据的内容是多样的。代表工具包括HDFS文件系统、HBase列数据库等。

3. ETL

在数据采集时,要对这些海量数据进行有效的分析,还要将这些来自前端的数据导入到一个集中的大型数据库,或者分布式存储集群,并且在此基础上做一些简单的清洗和预处理工作。大数据时代的ETL面临的挑战,主要是导入的数据量大,每秒的导入量经常会达到百兆字节甚至千兆字节级别。另一方面,在大数据平台完成计算、分析和挖掘后,生成的结果通常是比较小的,为了做可视化展示或与其他业务系统交互,可能需要将其再导入到关系型数据库中。典型的ETL工具包括Sqoop、DataX等,可以满足不同平台的数据清洗、导入导出等需求。

4. 数据计算

大数据计算主要体现在数据的快速统计与分析上。统计与分析主要利用分布式数据库或者分布式计算集群来对存储于其内的海量数据进行普通的分析和分类汇总等,以满足大多数常见的分析需求。常见的工具包括MapReduce分布式并行计算框架、Spark内存计算模型、Impala大数据交互查询分析框架等。

5. 数据分析与挖掘

大数据的数据挖掘与传统的数据挖掘方法也存在一定的差异。首先,在大数据平台

下，数据的体量对挖掘的时效性提出了更高的要求。其次，数据的体量和多样性对模型的绝对计算精度要求降低，可以通过相对计算精度的提升在全样数据上获得更好的计算精度。最后，大数据平台下的数据挖掘可以没有什么预先设定好的主题，主要是在现有数据上面进行基于各种算法的计算，从而起到预测的效果，实现一些高级别数据分析的需求。常用的工具包括 Mahout、MLlib 等数据挖掘和机器学习工具。

6. 数据可视化

对于数据分析，最困难的一部分就是数据展示，解读数据之间的关系，清晰有效地传达并且沟通数据信息。大数据可视分析旨在利用计算机自动化分析能力的同时，充分挖掘人对于可视化信息的认知能力优势，将人、机各自的强项有机融合，借助人机交互式分析方法和交互技术，辅助人们更为直观和高效地洞悉大数据背后的信息、知识与智慧[8]。大数据时代数据的来源众多，且多来自异构环境。即使获得数据源，得到的数据的完整性、一致性、准确性都难以保证，数据质量的不确定性问题将直接影响可视分析的科学性和准确性。数据可视化已经融入到大数据分析处理的全过程当中，逐渐形成了基于数据特点、面向数据处理过程、针对数据分析结果等多方面的大数据可视分析理论[9]。典型的可视化工具或组件包括 D3.js、ECharts 等。

大数据挖掘处理基本流程及相应的工具如表 1-5 所示。

表 1-5 大数据挖掘处理基本流程

序号	环节名称	工具名称
1	数据采集	Flume、Kafka、Scribe 等
2	数据存储	HDFS、HBase、Cassadra 等
3	ETL	Sqoop、DataX 等
4	数据计算	MapReduce、Storm、Impala、Tez、Presto、Spark、Spark Streaming 等
5	数据分析与挖掘	Mahout、MLlib、Hive、Pig、R 语言等
6	数据可视化	D3.js、ECharts 等

1.5 大数据时代“互联网+”的未来：智能互联

本节展望大数据时代“互联网＋”的未来，为读者介绍智能互联的发展方向。

2015 年 3 月 5 日上午，十二届全国人大三次会议上，李克强总理在政府工作报告中首次提出“互联网＋”行动计划。“互联网＋”代表一种新的经济形态，即充分发挥互联网在生产要素配置中的优化和集成作用，将互联网的创新成果深度融合于经济社会各领域之中，提升实体经济的创新力和生产力，形成更广泛的以互联网为基础设施和实现工具的经济发展新形态。“互联网＋”行动计划重点促进以云计算、物联网、大数据为代表的新一代信息技术与现代制造业、生产性服务业等的融合创新，发展壮大新兴业态，打造新的产业增长点，为大众创业、万众创新提供环境，为产业智能化提供支撑，增强新的经济发展动

力，促进国民经济提质增效升级。

在该行动计划发布后，各方纷纷做出了积极响应，并结合各个应用领域进行了深度解读。其中，《中国新闻周刊》对“互联网＋”做出了深度解读，认为“互联网＋”可以看成是蒸汽机、电和互联网之后的又一次社会革命。蒸汽机、电和互联网引发了人类社会的三次工业革命。“互联网＋”也将导致新的一次工业革命。“互联网＋”这样一个深度互联的方式影响到社会的方方面面。例如政务、民生、交通、教育、医疗、金融、媒体和汽车等各行各业都能够通过“互联网＋”实现行业上网。大数据是颠覆性的技术，会使很多产业受到影响，上一次颠覆性技术出现的时候是互联网的时代，当时对各个行业都产生了影响，受到互联网冲击最大的就是零售业，在美国就是亚马逊和易贝。在“互联网＋”和人工智能蓬勃发展的今天，我们面临新的机遇与挑战，结合大数据智能分析技术的应用和案例遍地开花：

- **互联网＋政务**。截至2014年底，各级政府已经在微信上开通了近2万个公众账号，面向社会提供各类服务；武汉交警通过微信服务号可在60秒内完成罚款收取，此项功能全年可为武汉驾驶员窗口缴罚节省时间达140万小时，节约警力300人。2016年12月29日，广州市国家税务局、广州市地方税务局与腾讯公司签署了“互联网＋税务”合作备忘录，在互联网＋办税服务、互联网＋电子发票、互联网＋数据应用、互联网＋税企共治、互联网＋征管改革等多领域展开合作。“广州税务企业号”也于当天正式上线。这是全国首个“互联网＋税务”应用案例，广州成为全国第一个实现国、地税线上业务合并的试点城市。
- **互联网＋民生**。已经有91万广州市民通过微信上的“城市服务”入口获得医疗、交管、交通、公安户政、出入境、缴费、教育、公积金等17项民生服务。除广州外，深圳和佛山也已成为微信智慧城市，武汉和上海也将马上加入。
- **互联网＋交通**。滴滴打车等打车软件服务的出现，创造了12.1%的就业新机会。96.5%的司机在从事专车服务以后每月收入有所提升，其中，39.5%的司机有30%以上的收入提升。2016年7月28日，《网络预约出租汽车经营服务管理暂行办法》出台，以互联网技术为依托构建服务平台，接入符合条件的车辆和驾驶员，通过整合供需信息，提供非巡游的预约出租汽车服务。该办法规范了网约车市场，借助“互联网＋”和大数据手段帮助乘客和司机进行有效匹配，避免了空驶和疲劳驾驶，既方便了人们用车，又保护了环境。
- **互联网＋教育**。以慕课、翻转课堂等为手段的在线教育，有助于改善教育资源分配不均等现状，让每个人以更低成本获得更适合自己的学习资源。目前，腾讯已与超过5000家教育机构合作开设腾讯课堂，面向中小学、大学、职业教育、IT培训等多层次人群开放。每周上课人数超过7万人，课程总数达3万多门。
- **互联网＋医疗**。截至2016年12月，微信公众平台上完成认证的医院公众号接近2万个，近1亿患者通过微信公众号获取专业的医疗服务和资讯；国内1200多家三甲医院中，超过80%都开通了微信公众号，60%开通了就诊服务；通过微信支付，全国数千家医院每天为近50万患者提供便捷的支付服务，节约排队等候时间；腾讯在线挂号服务平台已覆盖了全国156个城市的1443家医院。

- **互联网＋金融**。2016年1月初，全国首家互联网银行“微众银行”开业，李克强总理敲下回车键，卡车司机徐军就拿到了3.5万元贷款。2016年12月22日，中信银行在京隆重发布金融同业合作互联网服务平台“中信同业＋”。
- **互联网＋媒体**。2015年春晚微信“摇一摇”送出5亿元红包，全球185个国家的用户摇了110亿次，最高峰时一分钟有8.1亿次“摇一摇”互动。
- **互联网＋汽车**。业界估计，到2020年，全球75%的汽车，也就是9200万辆汽车都将能够接入互联网的硬件设备。腾讯推出的车载智能硬件“路宝”盒子不仅可以提供实时路况、街景、测评驾驶，还可以对故障进行智能检测。

“互联网＋”跨行业、跨领域的深度互联将会是未来发展的一个热点，我们暂且称之为“互联网＋＋”，其中工业物联网和工业4.0受到了人们的广泛关注。工业互联网是将互联网中人与人之间的沟通延续到人与机器的沟通以及机器与机器之间的沟通。工业4.0实现了在三元空间中人、机器和信息之间的相互连接，其中包括三个层次：一是智能工厂，这是一个实体、静态的概念；二是智能生产，这是工厂内部生产管理的一个过程；三是智能物流，实现了智能工厂之间的互联。

- 从**计算机发展史**来看“互联网＋＋”，首先经历了单台计算机阶段，通过通信实现计算机之间连接的初级互联阶段，然后通过联网上线迈入了互联网时代。近年来，随着人机交互的发展，人的活动更多、更深入地介入整个互联网计算系统之中，也就有了现在的云计算时代。
- 从**工业发展史**来看“互联网＋＋”，传统工业通过计算机通信实现了工业信息化，通过实体经济上网、O2O等实现了工业信息化到智能工厂的转变。这些智能工厂在网络上互联之后，它们的交互可能带来更进一步的深度互联。这时候，管理、生产和服务都是一种在未来智能互联的网络上的行为。未来社会生产方式的发展与计算机互联计算的发展有着结构上的相似性。

云计算、大数据、“互联网＋”究竟能够带来社会上多大的变革呢？我们来看一看“1＋1远大于2”的经典案例：在20世纪60年代，美国政府提出了“中央数据银行”计划。这个计划旨在以公民为单位，为每人建立一个数据档案，包括其教育、医疗、福利、犯罪、纳税等各个方面的数据记录。然而，美国国会并没有批准这一计划。为什么？因为这些数据分散在每一个行业、每一个部门，还是一个个的数据孤岛，它的价值没有那么大，但是一旦汇聚在一起就附加了很大的价值，在大数据精准分析和挖掘面前，每个公民就是一个透明人，这些信息或分析结果一旦泄露，将会造成极其严重的后果，无法保证公民的隐私权。在“9·11”事件之后，美国政府在《国土安全法》中重提“中央数据银行”计划，而且提出了一个更为响亮的名字：万维信息触角计划(Total Information Awareness)，旨在追踪恐怖分子的“数字脚印”，以解决安全问题。和1965年相比，2002年，人类的计算模式已经稳步进入了个人型计算阶段，商务智能的各项技术都已经很成熟。因此，除了将数据联接、集中到一起，提高管理、查询和统计的效率之外，万维信息触角计划还有了新的内容——数据挖掘，将挖掘出的信息转化为知识并付诸行动。万维信息触角计划公开之后，引起了隐私、民权保护团体的强烈反响。美国国会最终终止了这一计划，理由是万维信息触角计划就像原子弹一样可以改变世界。也就是说，如果把不同方面的数据融合到一起，这一过

程有可能会带来像原子核聚变一样的效果。万维信息触角计划被终止了，但是这方面的研究工作却没有停止。

再看一看智能互联的聚变效应。首先是传统的互联，有一句话对互联有一个很形象的说明：要致富先修路。以前修的是公路，现在要修的是网络、光纤道路，是在科技上的互联。多学科的交叉成为当前创新的源泉。自然界中有广泛的互联，有各种各样互联的群体，如蚁群、蜂群、狼群。一只狼不可怕，如果有一群狼，战斗力就变得很可怕。云计算和大数据所带来的互联是信息系统之间的互联，物理空间与信息系统的互联，以及在三元空间中人与人、人与信息系统、人与物理环境以及人与物理环境和信息系统之间深度的、广泛的互联。

1.6 本书架构

本书共 11 章，除第 11 章以外，全书内容可以分成 3 个部分，整体篇章架构如图 1-12 所示。第 1 部分是大数据挖掘与应用的导论部分。首先展现了在大数据挖掘和分析中对各种各样的应用进行有效数据模式发现的方法，介绍了有效的和应用广泛的大数据分析工具的技术和分布式并行计算平台，同时对数据挖掘的基本概念和发现知识的模式类型技术进行了概要介绍；其次，由于好的分析结果必须由数据质量决定，因此从数据的类型入手，说明了保证数据质量的数据准备和预处理方法；最后详细介绍了一些普通常用的可视化技术，为进一步可视化展现海量数据中隐藏的现象提供了帮助。第 2 部分为大数据挖掘常用方法的介绍，包括关联分析方法、分类方法、聚类方法、深度学习，以及目前流行的用于大数据分析的 R 语言基本知识和编程方法介绍。第 3 部分是进阶应用部分，分别探讨了目前最有代表性的大数据分布式存储与并行计算的软件框架 Apache Hadoop 的编程基础、基于 Apache Storm 的分布式实时计算方法和基于 Spark Streaming 的分布式实时计算的基本技术，同时对并行计算算法的基本知识和概念做了介绍，通过 3 个完整的实际案例给出了 MapReduce 平台下数据分析的具体方法和步骤，使大家能够亲身感受在大数据平台下进行完整数据分析的过程。

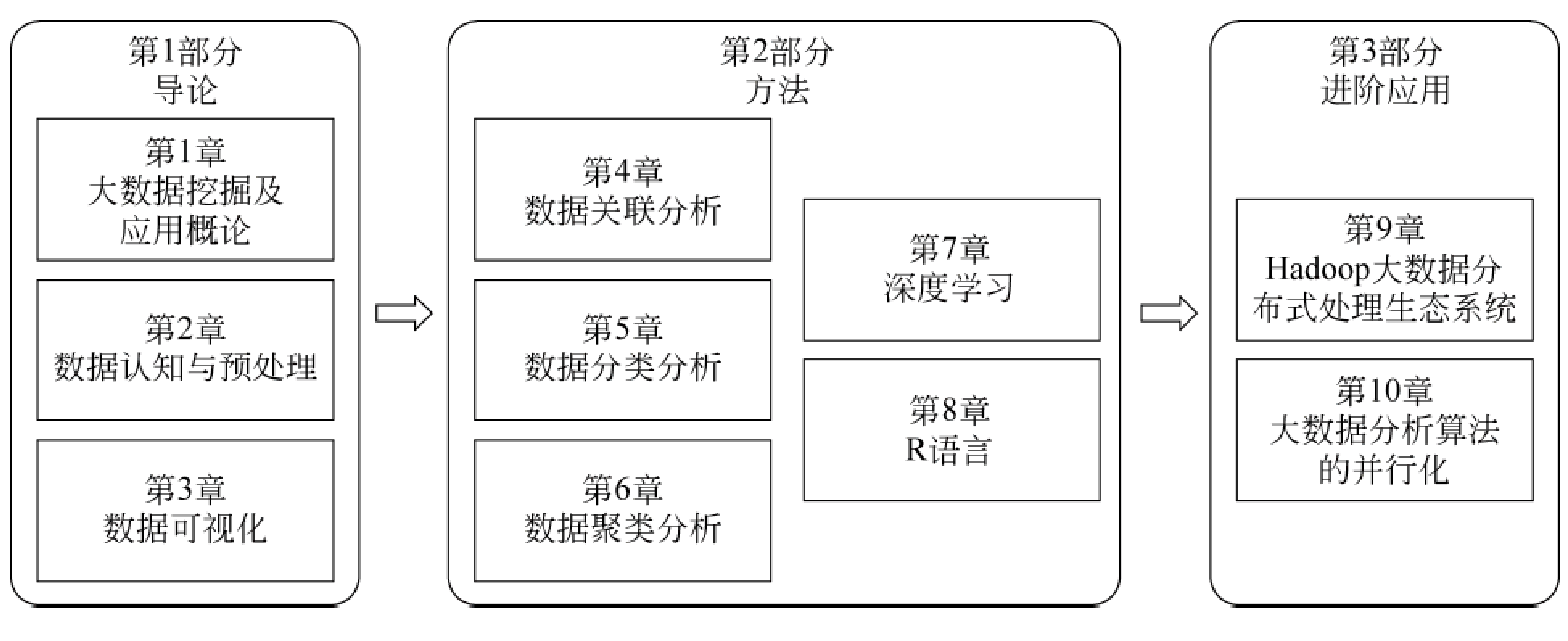


图 1-12　本书篇章架构图(第 1～10 章)

1.7 小　　结

人类正在迈入后信息时代的“三元空间”世界。信息(数据)成为物质和能量之外的新资源。云计算是新生产力工具,谁先掌握并运用好了,谁就在社会经济的发展中率先拥有了主动权。通过本章的学习,希望为读者在云计算、人工智能背景下的大数据智能分析处理领域打开一扇窗户,从宏观世界、现实社会、基础理论和技术方法上初步认识大数据智能分析与处理。

- **MapReduce 计算框架**的计算效率低,任务出错需要重做,编程接口简单,计算模型受限,适用场景包括文本处理、Log 分析、机器学习。
- **Spark 计算框架**的计算效率高,基于 RDD 的 Lineage 实现容错,内存计算通用性好,更适合迭代式任务,适用场景包括迭代式离线分析任务、机器学习。
- **Dryad 计算框架**的计算效率较高,任务出错需要重做,针对 Join 进行了优化,允许动态优化调度逻辑(修改 DAG 拓扑),适用场景包括机器学习、微软技术栈。
- **GraphLab 计算框架**的计算效率较高,基于检查点技术实现容错,机器学习图计算专用框架,适用场景包括机器学习、大图计算。
- **Storm 计算框架**的计算效率高,基于 Worker 重启或分配到新机器实现容错,任务出错时自动重做,通用性好,消息传递可靠,支持热部署,主节点可靠性差,适用场景包括通用的实时数据分析处理。
- **S4 计算框架**的计算效率高,基于检查点技术实现部分容错,通用性较好,通信在 TCP 和 UDP 之间权衡,持久化方式简单,适用场景包括实时广告推荐、容忍数据丢失。
- **Samza 计算框架**的计算效率高,任务出错重做,可扩展性好,兼容流处理和批处理,适用场景包括在线和离线任务相结合的分析与处理。
- **Spark Streaming 计算框架**的计算效率低,基于 RDD 和预写日志(Write Ahead Logs)实现容错,通用性好,容错性好,通过设置短时间片实现实时处理,但应用较受局限,适用场景包括历史数据和实时数据相结合的分析。

1.8 习　　题

1. 大数据处理的 5V(Volume、Velocity、Variety、Veracity、Value)特点:其含义分别是________、________、________、________、________。

2. 谷歌大数据技术的三大法宝包括________、________、________。

3. 云计算是基于互联网的相关服务的增加、使用和交付模式,通常涉及通过互联网来提供________且经常是________的资源。

4. 简述大数据处理的一般性框架。

5. 简述大数据处理的一般性步骤流程和方法。

6. 简述离线计算、迭代式计算和流式计算的主要代表技术及其特点。

7. 简述图灵计算和云计算的异同。

8. 结合现实，谈谈你对“互联网+”的理解。

1.9 参考文献

[1] 国务院. 国务院关于促进云计算创新发展培育信息产业新业态的意见：国发[2015]5号. 2015.

[2] 全国信息技术标准化技术委员会大数据标准工作组，中国电子技术标准化研究院. 大数据标准化白皮书：2016版. 2016.

[3] 中国信息通信研究院. 大数据白皮书. 2016.

[4] 阿莱克斯·彭特兰. 智慧社会：大数据与社会物理学. 汪小帆，汪容，译. 杭州：浙江人民出版社，2015.

[5] 赵晟，姜进磊. 典型大数据计算框架分析. 中兴通讯技术，2016，22(2)：14-18.

[6] 维基百科. 大数据. https：//zh. wikipedia. org/wiki/.

[7] Spark 峰会. https：//spark-summit. org/2015.

[8] 任磊，杜一，马帅，等. 大数据可视分析综述. 软件学报，2014，25(9)：1909-1936.

[9] 袁晓如. 大数据可视分析. 第一届科学大数据大会，北京，2014.

[10] 陈为，沈则潜，陶煜波，等. 数据可视化. 北京：电子工业出版社，2016



数据认知与预处理

直接从各种设备或系统等平台上采集的原始数据总是来自异种数据源，往往有大量的噪声、离群点和冗余存在，大体上来说，这样都是不完整、不一致的脏数据，数据质量太差，无法直接进行数据挖掘，或挖掘结果不如人意。比如遥感数据，可能是同时获取于几颗不同分辨率的卫星图像，还可能包含多个时间点采集的数据，因此它具有多源性，而每一个源的数据格式和表示都不相同。再比如进行网络入侵检测的判断，要想判断某一个 HTTP 请求是正常的还是恶意的，势必首先要把这些数据抓取下来。这一步好办，可以用一个探针程序部署在网关上，把所有经过这个网关的 HTTP 请求全部抓取下来并存储，但是程序抓取的都是一些乱七八糟的字符，于是首先要对它进行解码，然后定义好一定的格式将它们存储起来，其实这还远远不够，所获取的数据不一定干净，会有噪声，可能还会有缺失。因此一个完整的数据挖掘系统必须包括数据预处理模块，它以任务为导向，以专业知识为指导，用全新的"任务模型"来组织原始数据，摈弃一些与挖掘任务不相关的属性，填补缺失值，过滤那些会影响分析结果的噪声点，为数据挖掘算法提供完整、一致、准确、有效的数据，降低挖掘计算涉及的数据处理量，提高挖掘效率，高质量地、准确地发现知识。好的数据预处理有助于保证挖掘数据的正确性和有效性，另一方面也能通过对数据格式和内容的调整，使数据更符合挖掘的目标。

总之，对不理想的原始数据进行有效的归纳和预处理，已经成为数据挖掘系统实现过程中的关键问题。数据预处理不仅包含数据清理、数据集成、数据变换、数据规约等技术，还包括对数据进行统计描述，以考察数据值的分布情况，洞察数据的相似性等，数据可视化也是数据预处理技术的一个分支，这部分内容将在第 3 章进行介绍。这些数据处理技术在数据挖掘之前使用，大大提高了数据挖掘模式的质量，降低实际挖掘所需要的时间。本章首先从什么是数据分析入手，给出评估高质量数据的指标(2.1 节)，然后对数据由什么类型的属性或字段组成，每个属性具有何种类型的数据值，是离散属性还是连续属性进行描述(2.2 节)，对数据中心趋势和离散趋势的绘制方法进行分析(2.3 节)。度量数据间的相似性是数据挖掘重要的基础理论之一，本章探讨多种评估相似性和相异性的方法(2.4 节)，进而考察数据挖掘前必要的数据清理、数据集成、数据归约和数据变换方法(2.5 节)，最后介绍目前常用的一些用于数据统计分析和预处理工具(2.6 节)，并在 2.7 节用一个案例对 SPSS 工具的使用基础进行介绍。

2.1 数据分析的定义和流程

人们每一次点击鼠标，每一次敲击键盘，都可能在万里之外的某台服务器里产生一些数据。以前，我们存储数据但是并没有充分发挥出数据的价值，如今，数据分析技术的发展却使这些数据变成了潜力巨大的金矿。那么什么是数据分析？数据分析（data analysis）是指用适当的统计分析方法对收集来的大量数据进行分析和解释，提取出有用的信息形成结论，从而对数据加以详细研究和概括总结的过程。数据分析的数学基础早在20世纪初就已确立，但是直到计算机出现才使其实际操作成为可能，并使其得以推广。所以，数据分析是数学与计算机科学相结合的产物。数据分析的标准流程分为以下几步：数据分析问题的理解、相关数据的理解、相关数据的准备、分析模型的建立、分析模型的评估、分析结果的部署。

本节给出数据分析的定义和流程的相关知识。2.1.1节将讨论如何理解和描述数据分析问题，2.1.2节讲解数据获取与准备的相关知识，数据质量评估的相关知识将在2.1.3节提供。

2.1.1 如何理解和描述数据分析的问题

下面给出一个有意思的数据分析案例。美国明尼苏达州一家塔吉特百货公司的门店被客户投诉，一位中年男子指控塔吉特将婴儿产品优惠券寄给他的女儿——一个高中生。但没多久他却来电道歉，因为女儿经他逼问后坦承自己真的怀孕了。塔吉特就是靠着分析用户所有的购物数据，然后通过数据分析模型的分析评估获取顾客想要购买的商品的信息。类似的案例多不胜数，处处都展现着数据分析的魅力。

那么如何理解和抽象出一个待分析的数据分析问题模型呢？可以从以下几个方面进行考察：

（1）理解待分析的问题。理解问题的背景，确定待解决问题的目标，制定该问题解决成功的标准。

（2）考察待分析问题的当前形势。确定本问题所面临的资源需求、假设和约束，探讨结果在运用中可能带来的风险并制定应急方案，评估数据分析产生的成本和收益。

（3）确定待分析问题的数据分析模型。明确数据分析模型应该取得的目标，确立数据分析模型成功的评价标准。

（4）制定实施该问题的数据分析步骤。制定并明确项目实施计划，评估并选择适合的数据分析平台和技术。

下面给出一个如何理解和抽象出电信客户流失的数据分析问题模型的案例。在电信行业，客户流失的意思是客户从某家电信公司退出而转到另一家具有竞争关系的电信公司。在电信行业通常争取一个新客户的成本是保留一个老客户所需成本的5倍，如果能使客户流失率降低5%，企业利润可以增加25%～85%，因此在电信行业防止客户流失是企业极其重视的问题，那么如何理解和抽象出对客户流失进行数据分析的问题模型呢？

(1) 首先对问题进行理解，例如什么是流失，流失如何定义。连续欠费不缴、号码长期不用等都属于流失的范畴，数据分析成功的目标和标准就是找到流失和哪些因素相关，流失客户的特征是什么。

(2) 分析当前客户对电信使用情况的形势。确定客户流失可能会与顾客年龄、性别、收入、职业、话费水平、话务质量等哪些因素有关，显然，如果数据分析结果有效，能够大大增加企业利润。

(3) 确定数据分析模型。要寻找那些流失量比较大的客户群，对客户进行分析的模型应该用什么？这里显然应该对客户进行聚类分析，将具有相同特征的客户聚簇，从而及早发现问题，避免客户流失。尤其对那些流失性大的高价值客户，建立模型规则集和模型评估方法，从中选出最优模型，更有利于发现那些易于流失的客户群特征。

(4) 制定实施电信客户流失数据分析的计划，并根据数据量的大小确定数据分析平台和分析算法。

2.1.2 数据获取与准备

俗话说“巧妇难为无米之炊”，同样，要进行数据分析，当然先要有数据。根据不同的分析目的，需要获取的数据的种类、形态也不尽相同。一般来说，获取数据的途径主要可分为以下几种。

1. 数据库

数据库(database)是数据分析中最常见、最丰富的数据源，也是数据分析中所需数据的最基本承载形式。依据所处理的数据形式，数据库可以分为多种类型：

(1) 关系数据库(relational database)。即采用关系模型作为数据组织方式的数据库，其特点在于它将所有的数据对象存储在一个以行和列方式组织的二维表中。

(2) 事务数据库(transactional database)。是一种特殊的关系数据库，其中存放的每个数据(即记录)代表不同的交易事务，如顾客的一次购物、一次航班订票或一次网页点击等。

(3) 多媒体数据库(multimedia database)。这种数据库一般存放图像、音频和视频类等数据。

(4) 遗留数据库(legacy database)。是企业早期开发遗留下来的数据，可以是关系型、面向对象型或层次型等的不同异构数据库。

数据库中数据的来源非常丰富，可以是在互联网上抓取的数据，也可以是通过各种不同传感器或其他设备得到的实测数据，还可以是通过调查报告等方式人工收集的数据。

2. 数据仓库

数据仓库(data warehouse)是进行数据分析最常用的数据来源。建立数据仓库的主要目的是为工商企业主管提供一种分析工具，以便他们能够系统地组织、理解和使用数据进行决策。它和数据库的主要不同表现在操作方式和目的上。对数据库的操作称为联机事务处理(Online Transaction Processing, OLTP)，主要功能是对实时数据进行日常操

作，比如增加、删除、更新、查询等；而对数据仓库的操作称为联机分析处理（Online Analytical Processing，OLAP），主要功能是对长期保存的历史数据进行复杂查询，进行的主要是读操作，最终提供决策支持。

数据仓库中的数据来源主要是通过对大量异构的数据库中的数据进行清洗、预处理以及转换等方法加以集成，重新组织到一个语义一致的数据存储中。

3. 文件

文件（file）是获取数据分析所需数据的主要来源，它包括后缀为 doc、xls、txt 等的普通数据文件，从这些文件中快速获取的数据容易理解，但是要处理成与数据分析中其他数据一致的格式比较困难。

数据获取的方式其实还有很多，这里只介绍了常用的 3 种方式，一般往往需要根据实际情况来选择最适合的方式。原始数据获取直到可以用来进行数据分析，中间还要经过几个步骤，数据获取只是其中的第一步，称为加载（loading），还需要经历清洗（cleaning）和集成（integration），然后才能进行分析。通常只有用来分析的数据质量足够高，才能保证分析结果更有效。那么如何评价获得的数据质量如何呢？下面介绍数据质量的评估标准。

2.1.3 数据质量评估

获取的数据如果能满足其应用要求，那么就说它是高质量的。但是获取的数据往往很难达到应用要求。例如，需要收集某网站商品价格信息，但是发现有些字段根本没有值，我们就说这个数据是不完整的。也可能获取的价格信息本身就是错误的，那么这个数据就是不准确的。另外，可能获取的用于商品分类的编码存在差异，那么这个数据是不一致的。

一般来说，准确性、完整性、一致性、时效性等指标是评价数据质量最常用的标准。

1. 准确性

准确性是指数据记录的信息是否存在异常或错误。最常见的数据准确性问题就是乱码，过大和过小的数据往往也是不准确的。数据质量的准确性问题可能存在于个别记录中，也可能存在于整个数据集中。

2. 完整性

完整性是指数据是否存在缺失，缺失可能是整个记录数据，也可能是记录数据中某个关键字段的缺失。不完整的数据价值会大大降低，因此，完整性也是最基础的数据质量评估标准。

3. 一致性

一致性是指数据是否遵循了统一的规范，数据集是否保持统一的格式。数据的一致性主要体现在数据记录是否规范和数据是否符合逻辑。例如，手机号码一定是 11 位，中

国人姓名一定是汉字组成的这类规定就属于规范。逻辑指的是数据间存在着特定的逻辑关系，如对网页访问的 PV（访问量）一定大于等于 UV（独立访客量）。

4. 时效性

时效性是指数据从产生到得到分析结果的时间间隔。如果数据分析周期加上数据建立的时间间隔过长，就可能导致分析结果失去应有的意义。

2.2 数据类型

我们获取的数据集都由一个一个数据对象组成，每一个对象都代表一个实例。例如某人手机里的通讯录数据，其中的每条数据对象代表一个人的联系方式，而这条联系方式中有一部分表示姓名，一部分表示电话号码，还有一部分表示这个人的住址等等。其中，每一条数据可以称为数据集的一个样本、实例、数据点或对象。而每一条数据要用不同特征描述出来，特征也可以称为属性，如上例中的姓名、电话号码、住址都是描述数据用到的特征。如果数据存放在数据库中，则行对应于数据对象，列对应于数据属性。

本节将对属性进行定义，同时考察属性可以具有的不同类型。首先对属性这一概念进行描述，然后分别讨论标称属性、二元属性、序值属性以及数值属性。

2.2.1 属性的定义

属性（attribute）是一个字段，表示数据对象的一个特征。对象与属性是不可分的，没有属性的对象是不存在的，因为对象不用属性来进行描述也就不能称之为对象了，而属性如果不用来描述对象，也就没有意义。

在文献中，属性、字段、维（dimension）、特征（feature）和变量（variable）可以表示相同的意义，在不同场合它们总是互换使用。术语“维”一般用在数据仓库中，“特征”较多用于机器学习领域，而统计学中则更多地使用术语“变量”。数据挖掘和数据库领域则一般使用术语“属性”或“字段”。通常把描述对象的一组属性称作这个对象的属性向量。

属性的取值范围决定了属性的类型，通常可以分为两大类。一类是定性描述的属性，其中可以划分为标称属性、布尔属性和序值属性。定性属性不具有数的大部分性质，即使用数（即整数）表示，也应当像对待符号一样对待它们；另一类是定量描述的属性，即数值属性，定量属性用数表示，并且具有数的大部分性质，定量属性可以是整数值或连续值。下面将一一介绍以上各类属性的定义。

2.2.2 标称属性

标称属性（nominal attribute）取值仅仅只是一些不同的符号或事物的名称，每个值提供了足够的信息以区分对象。这些值不具有有意义的次序，在计算机科学中，可以将这些值看作是枚举的（enumeration），例如邮政编码、学生 ID、头发颜色等。

例 2-1 可以用名称、种类、颜色、是否成熟、品级等属性来描述水果类的数据对象，

如表 2-1 所示。

表 2-1 对象苹果属性组

名称	种类	颜色	是否成熟	品级
苹果	核果类	红色	是	优

名称的值可能是梨、苹果、桃子等,种类的值可能是浆果类、核果类、柑橘类等,而颜色的值可能是红色、青色、黄色等。在这个例子中,它们的值分别是苹果、核果类、红色,这些值表明了其所描述的对象苹果的属性,这就是一般意义上的标称属性。为了方便,也可以用数表示这些标称属性。例如,可以定义 1 表示苹果,2 表示梨,3 表示桃子,那么苹果在名称属性上取值为 1。不过即使用数字来表示这些属性的值,也不能定量地使用这些值,例如不能用苹果(1)减去梨(2),这样做是没有意义的。一般情况下,不能求这些值的均值、中位数,但是可以求出该属性下最常出现的值,这个值称为众数(mode),是一种中心趋势度量,2.3.1 节将介绍数据的中心趋势度量。

2.2.3 二元属性

二元属性(binary attribute)是只有两个可选值的属性。该类属性只有 0 和 1 或者 True 和 False 两个状态,一般 0 表示"否",1 表示"是"。在表 2-1 中,是否成熟这一属性就是二元属性,可以用值为 1 表示成熟,值为 0 表示不成熟。如果一个二元属性的两种状态有相同的权重,就说这个二元属性是对称的;如果两种状态权重不同,则这个二元属性是非对称的。非对称的二元属性经常用在医学领域,表示某种医学指标的阴性或阳性,为了方便,通常用 1 表示权重较大的状态。

2.2.4 序值属性

序值属性(ordinal attribute)的值提供了足够的信息确定数据对象之间的序,但是值之间的差是未知的。在例 2-1 中,品级属性的取值有优、良和差。这 3 个值有一个先后次序,表示了苹果的品级好坏,属于序值属性。在 2.5.2 节中可以看到,数值属性可以转换为序值属性,主要是通过把值域进行离散化得到。序值属性可以定义众数、中位数或百分位数,但是不能定义均值。

2.2.5 数值属性

数值属性(numeric attribute)是最常用的一种数据类型,它是可度量的,用整数或实数值表示,它定量地描述对象。一些文献中将数值属性又划分为区间标度或比率标度属性。其中区间标度(interval-scaled)属性存在测量单位,值之间的差是有意义的,因此可以比较和评定这种属性值之间的差。对于区间标度属性,既可以计算中位数和众数,也可以计算均值和标准差等。比率标度(ratio-scaled)属性是具有固定零点的属性,它的差和比率都是有意义的,同样可以计算其均值、中位数及众数等。

2.3 数据的统计描述方法

在对数据进行分析之前，把握数据的全貌是至关重要的。基本的统计描述方法不仅可以用来识别整个数据集的性质和特点，发现数据集中的噪声或离群点，还能够对缺失的数据值进行补全。

本节讨论两类基本统计描述。一类是度量整个数据集合的中心趋势的方法，其中最常见的度量方法是均值、中位数、众数和中列数。这几种度量方法从不同角度描述了数据集的中心位置。另一类是度量整个数据集的离散趋势的方法，主要包括极差、分位数、五数概括、方差和标准差。这些度量方法从不同角度描述了数据集的离中趋势，对于识别离群点是非常有用的。本节首先讨论数据的中心趋势度量的相关问题，然后讲解数据离散趋势度量。

2.3.1 数据的中心趋势度量

中心趋势是指一组数据向某一中心值靠拢的倾向，测量中心趋势就是要寻找数据一般水平的代表值或中心值。

假设有一个表示年龄的属性 X，通过采集得到 X 的一组观测值 $x_1, x_2, \cdots, x_N$，这些值又称为属性 X 的数据集合。那么这一组观测值大部分落在何处呢？描述中心趋势的统计指标可以回答这个问题。下面介绍几种常用的中心趋势度量：均值、中位数、众数和中列数。

1. 均值

数据集中心最常用、最有效的数值度量之一是均值(mean)，均值有很多种，包括算术均值和几何均值等。算术均值公式如下：

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N} \tag{2-1}$$

例 2-2 对于属性 X，若观测值依次为 17，10，23，15，19，18，30，25，9，4，12，10，使用式(2-1)计算如下：

$$\bar{x} = \frac{17+10+23+15+19+18+30+25+9+4+12+10}{12} = \frac{192}{12} = 16$$

因此这组观测值的均值为 16。

权重(weight)是一个相对的概念，它反映了对应观测值的相对重要程度、意义或出现频率，也就是说要从若干个观测值中分出轻重来。如果对于属性集合 X，其每个值 x_i 有一个权重 ω_i 与之对应，这时，可以计算数据集的加权算术均值，计算公式如下：

$$\bar{x} = \frac{\sum_{i=1}^{N} \omega_i x_i}{\sum_{i=1}^{N} \omega_i} = \frac{\omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_N x_N}{\omega_1 + \omega_2 + \cdots + \omega_N} \tag{2-2}$$

虽然均值是描述数据集最有用的单个量，但是它并非总是度量数据集中心趋势的最佳方法。均值对极端值（如离群点）极为敏感，如果将上述属性 X 的第一个观测值 17 改为 100，这时算得的均值为 22.9。这显然没能正确反映数据集的中心，因为极端大的值 100 显著拉高了数据集的均值。为了抵消极端值的影响，可以使用截断均值（trimmed mean）。计算截断均值时，需要丢弃数据集中极端大和极端小的值（如去掉属性 X 观测值中的最大和最小值），或者分别丢弃高端和低端值的 2%，但应避免在两端截去太多，因为这样容易丢失有用的信息。

2. 中位数

大多数时候数据集合中的观测值分布是不均匀的，这样的数据集合称为倾斜数据集，一般会使用偏斜度来度量数据分布的偏斜方向和程度。均值不能很好地反映倾斜数据集的中心，这时候可以计算其中位数（median）或众数（mode）。中位数是有序数据集的中间值，它将数据较大的一半和较小的一半分开。

假定将数据集 X 的 N 个观测值按序排列。如果 N 是奇数，则中位数就是该有序集的中间值；如果 N 为偶数，则中位数不唯一，它是最中间的两个值之间的任意值，若 X 是数值属性，这时中位数可以计算中间两个值的平均值得到。

对于例 2-2 的属性 X 来说，其有 12 个观测值，观测值个数为偶数，因此中位数为 $(17+15)/2=16$。如果去掉该属性的最大观测值 30，则观测值个数为奇数，这时数据集的中位数为该组观测值有序排列后的中间值，即为 15。

当数据集的规模很大时，计算中位数开销很大。若观测值是数值属性，可以通过计算其中位数的近似值反映该数据集的中心趋势。首先将数据集的 x_i 值划分成区间，并且已知每个区间数据值的个数，称包含中位数的区间为中位数区间，此时可以使用以下公式，用插值计算得到整个数据集中位数的近似值：

$$\text{median} = L_1 + \frac{N/2 + \left(\sum \text{freq}\right)_1}{\text{freq}_{\text{median}}}\text{width} \tag{2-3}$$

其中，L_1 是中位数区间的下界，N 是整个数据集中观测值的个数，$\left(\sum \text{freq}\right)_1$ 是低于中位数区间的所有区间观测值的个数总和，$\text{freq}_{\text{median}}$ 是中位数区间中观测值的个数，width 是中位数区间的宽度，即中位数区间最小和最大值的差。

3. 众数

一组数据集中出现次数最多的值叫众数（mode），有时众数在一组数据集中不止一个。具有一个、两个或三个众数的数据集分别称为单峰（unimodal）、双峰（bimodal）或三峰（trimodal）数据集。有两个或两个以上众数的数据集统称为多峰（multimodal）数据集。如果数据集中每个数据值只出现一次，则该数据集没有众数。例 2-2 中 X 数据集的观测值众数为 10。

4. 中列数

中列数（midrange）是数据集最大值和最小值的平均值。对于例 2.2 的 X 属性，其中

列数可以计算为(30+4)/2=17。

偏斜度是对数据分布偏斜方向及程度的度量。数据集的分布有的是对称的，有的是不对称的，即呈现偏态，在大部分实际应用中，数据都是不对称分布的。在偏态分布中，当偏斜度为正值时，称为分布正偏或正倾斜，即众数位于算术平均数的左侧，小于中位数的值，如图 2-1(a)所示；当偏斜度为负值时，称为分布负偏或负倾斜，即众数位于算术平均数的右侧，大于中位数的值，如图 2-1(c)所示。可以利用众数、中位数和算术平均数之间的关系判断分布是左偏态还是右偏态。在完全对称的数据集中，均值、中位数和众数都是相同的中心值，如图 2-1(b)所示。

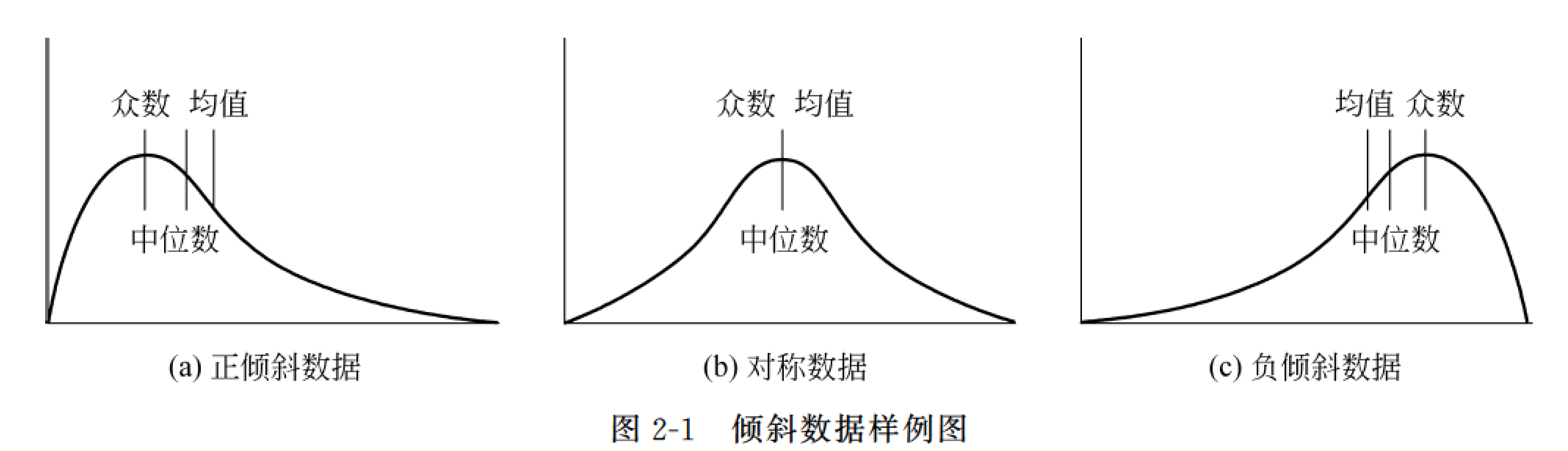


图 2-1 倾斜数据样例图

2.3.2 数据的离散趋势度量

为了把握数据的全貌，除了中心趋势度量外，还可以度量数据的离散趋势。离散趋势度量反映了数据集中的值远离其中心值的程度，因此也可以叫做离中趋势度量。离散趋势度量主要有极差、分位数、五数概括、方差和标准差等几种度量方法。

1. 极差与分位数

极差(range)又称全距，是指一组数据集观测值中的最大值和最小值之差。它只考虑了数据中的最大值和最小值，忽略了全部观测值之间的差异，因此极差反映的是一组数据集合中最大的离散程度情况。

将数据集中所有观测值按递增顺序排列，然后把数据划分成大小基本相同的连续集合，每隔一定距离取数据分布上的一个数据点，这个数据点就叫做数据集的分位数。假设将数据集划分为 k 个部分，则从数据集的起始点到终点总共可以取出 $k-1$ 个数据点，这些数据点称为 k 分位数。分位数(quantile)是统计学中的概念，它主要用于度量呈偏态的定量数据的离散趋势，常用的有四分位数、十分位数和百分位数。

这里主要介绍四分位数，其他以此类推。如果将有序数据集划分为 4 个间距相等的部分，每部分包含数据集中四分之一的数据，这时产生 3 个数据点，通常称它们为四分位数。四分位数给出了数据分布的中心、散布和形状的某种指示。第 1 个四分位数记作 Q_1，它处在数据分布 25%的位置；第 2 个四分位数处在数据分布的中心位置，也就是中位数；第 3 个四分位数记作 Q_3，它处在数据分布 75%的位置。

第 3 个和第 1 个四分位数之间的距离是叫做四分位数极差(IQR)，它是度量数据集散布程度的一种最简单的方法。可以用如下公式计算四分位数极差：

$$IQR = Q_3 - Q_1 \tag{2-4}$$

例 2-2 中的 X 属性总共有 12 个数据点，按递增顺序排列后，该数据集 3 个分位数点上的值分别是 10、15 和 19，即 $Q_3=19$，$Q_1=10$，所以四分位数极差 IQR=19－10=9。

2. 五数概括与盒图

前面已经探讨过数据分布的均值、中位数、众数、极差和四分位数等度量。但是对于倾斜数据，单一指标并不能很好地描述数据的完整分布情况，引入五数概括(five-number summary)可以更加完整地描述数据的分布情况。五数概括中的五数由中位数、四分位数 Q_1 和 Q_3、最大和最小观测值组成，一般按次序 Minimum、Q_1、Median、Q_3、Maximum 分别写出。

通常使用盒图(boxplot)来直观地对这五数进行可视化表示，盒图又被称为箱线图，其呈现形式如下：

(1) 盒的端点在四分位数上，下端点是 Q_1，上端点是 Q_3，盒的长度是四分位数极差 IQR。

(2) 中位数在盒内用横线进行标记。

(3) 盒外用两条虚线分别延伸至最小和最大观测值，这两条虚线又称为胡须。

五数概括中的最大和最小观测值不是数据集合中的最大和最小值，其中最小观测值的计算公式为

$$\text{最小观测值} \geqslant Q_1 - 1.5\text{IQR} \tag{2-5}$$

最大观测值的计算公式为：

$$\text{最大观测值} \leqslant Q_3 + 1.5\text{IQR} \tag{2-6}$$

当数据量适中时，还可以适当绘制出个别离群点。识别离群点的一般规则是：挑选大于最大观测值或小于最小观测值的点。盒图可以从直观的可视化角度比较若干数据集的分布情况，如图 2-2 所示。图中的盒图是用 MATLAB 绘制的，其中 5 个数据集是调用 randint()函数生成的 1～100 中的 5 组随机数据集，通过图 2-2，可以直观地比较这 5 组

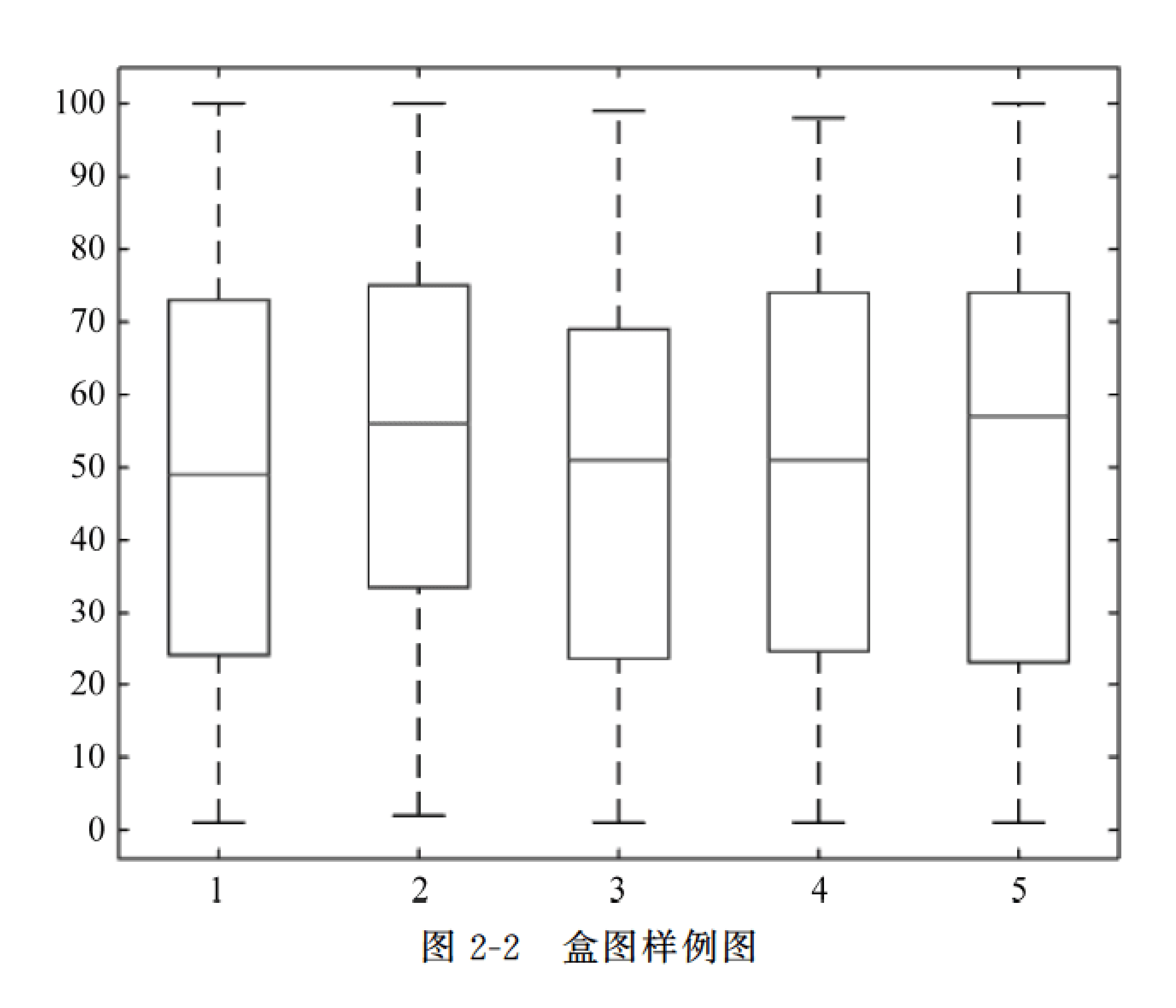


图 2-2　盒图样例图

数据的中位数、第1、3四分位数、最小和最大观测值的分布情况。

除了采用MATLAB绘制盒图，很多统计分析工具软件都具备绘制盒图的功能。

3. 方差和标准差

方差与标准差是观察数据散布情况的两个重要度量方法，它们可以指出数据分布的散布程度。标准差低表示数据观测值趋向于接近均值，标准差高表示数据观测值散布在更大的值域中，也就是说所有数据值的离散程度较大。

设属性 X 有 N 个观测值 $x_1, x_2, \cdots, x_N$，其方差可以用如下公式求得：

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{1}{N}\sum_{i=1}^{n} x_i^2 - \bar{x}^2 \tag{2-7}$$

其中 $\bar{x}$ 是观测值的均值，观测值的标准差为方差 σ^2 的平方根。

对于例2-2中的属性 X，其均值为16，N 为12，可以计算其方差和标准差分别为

$$\sigma^2 = \frac{1}{12}\times(17^2 + 10^2 + 23^2 + \cdots + 10^2) - 16^2 \approx 51.83$$

$$\sigma \approx \sqrt{51.83} \approx 7.20$$

众所周知，一个高质量数据集的采集，其得到的样本观测值应该很紧密地分散在样本真实值周围。如果不紧密，与真实值的距离就会大，相应离散度就大，那么准确性就不好，数据质量就不高。因此，离散度是评价数据质量好坏的最重要也是最基本的指标。由于方差是数据的平方，与观测值本身相差太大，人们难以直观地衡量，所以常用标准差来进行离散程度的度量。

一组合理的数据观测值一般不会远离均值，超过标准差的数倍。因此，标准差是度量数据集发散程度的很好的指标。

至此，关于数据集的中心趋势度量与离散趋势度量的探讨就告一段落。要更有效更直观地向用户展示数据，数据可视化是最佳的表达方法，数据可视化的一些方法将在第3章进行介绍，下一节将介绍数据对象的相似性与相关性计算方法。

2.4 数据对象关系的计算方法

上一节中的数据统计描述方法是围绕一组对象集合中某个属性的数据集合来使用的，本节对数据对象关系的计算方法围绕一组对象的多个属性数据展开。数据对象的计算通常都是指分析计算数据对象之间的相似性和相异性，相应的计算方法已经被许多数据挖掘技术所使用，如聚类、分类、离群点检测等。例如，电信企业为了降低其客户流失率，就需要通过对客户的聚类分析，也就是数据对象的相似性分析，发现那些易于流失的客户群的共同特征，根据其特征采取相应的解决办法。对象之间的相似性是指对象间相似程度的数值度量，数据的相似性也即邻近度，通常邻近度的值为非负值，取值范围在0和1之间，0代表完全不相似，1代表完全一致。在对相似性进行计算时，通常都是转换成计算对象之间的差异程度，即相异性，两个对象越相似，它们间的差异程度就越小，也就是相异性就越低，一般采用“距离”作为相异性的同义词。计算数据对象之间相异性的度量

方法很多，选择度量方法时，应观察对象特征的数据类型以及对象特征的总体分布状况等因素。2.4.1 节将对相似性计算方法进行详细讨论，2.4.2 节将讨论数据相关性的计算方法。

2.4.1 数据相似性计算方法

假设有 n 个对象，每个对象有 p 个属性，这些对象分别是 $X_1=(x_{11},x_{12},\cdots,x_{1p})$，$X_2=(x_{21},x_{22},\cdots,x_{2p}),\cdots,X_n=(x_{n1},x_{n2},\cdots,x_{np})$。其中 x_{ij} 是对象 X_i 的第 j 个属性值。对象通常是指关系数据库中的元组，也称数据样本或特征向量。在相似性计算中，总是通过数据矩阵(data matrix)这种数据结构将对象集合组织起来，例如用 $n\times p$(n 个对象，p 个属性)的如下矩阵存放以上 n 个数据对象：

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \tag{2-8}$$

其中每行表示一个对象。

一般用 $d(i,j)$来表示两个对象 i 和 j 之间的距离值，也就是相似度值，显然，$d(i,j)$越大，说明两者距离越大，相似性就越小。

相似性计算方法依赖于对象属性的数据类型，下面分别对标称属性、二元属性、序值属性以及数值属性等的相似性度量方法进行探讨。

1. 标称属性相似性

假设标称属性的状态数目是 M。两个对象 i 和 j 之间的距离可以根据对象属性的不匹配率来计算，公式如下：

$$d(i,j)=\frac{p-m}{p} \tag{2-9}$$

其中，p 是刻画对象的属性总数，m 是两个对象取值相同的属性数。

例 2-3 沿用 2.2 节中水果的例子，再增加几组数据，在表 2-2 中，考虑种类、颜色、是否成熟、品级 4 种属性。

表 2-2 数据对象水果样例表

名字	种类	颜色	是否成熟	品级
苹果	水果	红色	是	优
梨	水果	黄色	是	优
白菜	蔬菜	白色	是	良
桃子	水果	粉色	是	优

令 $p=4$，当对象 i 和 j 匹配时，$d(i,j)=0$；否则 $d(i,j)=(4-m)/4$ 。计算过程如下：

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix} = \begin{bmatrix} 0 & & & \\ \frac{1}{4} & 0 & & \\ \frac{3}{4} & \frac{3}{4} & 0 & \\ \frac{1}{4} & \frac{1}{4} & \frac{3}{4} & 0 \end{bmatrix}$$

还可以将标称属性通过编码方法转化为非对称二元属性来计算其相似性。假设标称属性有 M 种状态，对每种状态建立一个二元属性。对于有给定状态的对象，该状态对应的二元属性值置为 1，其余二元属性值都为 0。用这种形式编码的标称属性，可以用二元属性相似性计算方法来计算其相似性。

2. 二元属性相似性

二元属性只有两种状态：0 和 1，或者是 True 和 False。

二元属性的相似性度量分为对称的和非对称的两种情况。在分析对象相似性时，对称的二元属性的两个状态取值对相似性分析的贡献是一致的，而非对称二元属性的两个状态取值具有不同的权重，对相似性分析的贡献不一样。

为便于介绍二元属性的相似性度量方法，表 2-3 给出了两个对象所有的二元属性取值情况。

表 2-3　二元属性取值表

对象 i	对象 j		
	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

其中，q 是对象 i 和 j 都取 1 的二元属性数；r 是在对象 i 中取 1，在对象 j 中取 0 的二元属性数；s 是在对象 i 中取 0，在对象 j 中取 1 的二元属性数；t 是对象 i 和 j 都取 0 的二元属性数。对象所拥有的二元属性总数是 p，$p=q+r+s+t$。

如果对象 i 和 j 都是用对称的二元属性刻画的，则 i 和 j 的距离采用如下公式进行计算：

$$d(i,j) = \frac{r+s}{q+r+s+t} \tag{2-10}$$

如果对象 i 和 j 都是用非对称的二元属性刻画的，而且当属性值取 1 时权重最高，那么式(2-10)中的 t 认为是可被忽略的(即在属性总数中减去了两个对象都取 0 的属性值，因为这些属性在对象相似性分析中意义不大)，则 i 和 j 的距离计算公式如下：

$$d(i,j) = \frac{r+s}{q+r+s} \tag{2-11}$$

例 2-4 假设有两个对象 i、j，分别有 5 个非对称二元属性，如表 2-4 所示。

表 2-4 属性表

对象名	Attr-1	Attr-2	Attr-3	Attr-4	Attr-5
i	0	1	0	0	1
j	1	1	1	0	1

计算对象 i、j 的相异性。

因为 4 个属性都为非对称的二元属性，所以要用式(2-11)进行计算。先画出两对象的列联表，如表 2-5 所示。

表 2-5 对象 i 与 j 的列联表

对象 i	对象 j		
	1	0	sum
1	2	0	$q+r$
0	2	1	$s+t$
sum	$q+s$	$r+t$	p

根据式(2-11)可得

$$d(i,j)=\frac{0+2}{2+0+2}=\frac{1}{2}$$

所以，对象 i 和 j 的相异性为 1/2。

3. 数值属性相似性

在实际应用中，数值属性是最常用的一种属性类型。有很多方法来计算数值属性刻画的对象之间的邻近性。最常用的有欧几里得距离(又称为欧氏距离)、曼哈顿距离和闵可夫斯基距离等。

欧式距离(Euclidean distance)是高维空间中两点之间的距离，它计算简单，应用广泛，但是没有考虑属性之间的相关性，当多个属性参与计算时会影响结果的准确性，同时它对向量中的每个分量的误差都同等对待，一定程度上放大了取值范围较大的属性误差在距离计算中的作用。

令 $i=(x_{i1},x_{i2},\cdots,x_{in})$ 与 $j=(x_{j1},x_{j2},\cdots,x_{jn})$ 是两个被 n 个数值属性描述的对象，则对象 i 与 j 的欧氏距离定义为

$$d(i,j)=\sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+...+(x_{in}-x_{jn})^2} \tag{2-12}$$

欧氏距离虽然很有用，但也有明显的缺点。它将对象的不同属性之间的差别等同看待，这一点有时不能满足实际要求。例如，在教育研究中，经常遇到对人的分析和判别，显然描述个体的不同属性对于区分个体有着不同的重要性，而欧式距离却采用所有属性等同对待的方法。

曼哈顿距离(Manhattan distance)也称为城市街区距离，想象在曼哈顿要从一个十字

路口开车到另外一个十字路口，驾驶距离是两点间的直线距离吗？显然不是，除非你能穿越大楼。实际驾驶距离就是“曼哈顿距离”。其定义如下：

$$d(i,j)=|x_{i1}-x_{j1}|+|x_{i2}-x_{j2}|+\cdots+|x_{in}-x_{jn}| \tag{2-13}$$

欧氏距离和曼哈顿距离都有如下数学性质：

(1) 非负性。$d(i,j)\geqslant 0$，即距离是一个非负的数值。

(2) 同一性。$d(i,i)=0$，即对象到自身的距离为0。

(3) 对称性。$d(i,j)=d(j,i)$，即距离是一个对称函数。

(4) 三角不等式。$d(i,j)\leqslant d(i,k)+d(k,j)$，即从对象 i 到对象 j 的直接距离不会大于途经任何其他对象 k 的距离。

例 2-5 假设有两个被数值属性所刻画的对象，分别为 $A=(4,9,8)$，$B=(2,5,7)$，分别求出 A 与 B 的欧氏距离和曼哈顿距离。

对象 A、B 的欧氏距离：

$$d(A,B)=\sqrt{(4-2)^2+(9-5)^2+(8-7)^2}=\sqrt{21}$$

对象 A、B 的曼哈顿距离：

$$d(A,B)=|4-2|+|9-5|+|8-7|=7$$

切比雪夫距离（**Chebyshev distance**）**也称上确界距离**，是向量空间中的一种度量，两个点之间的切比雪夫距离可定义为其各坐标数值差中最大的差值。切比雪夫距离得名自俄罗斯数学家切比雪夫。切比雪夫距离也称为棋盘距离，国际象棋中，国王走一步能够移动到相邻的8个方格中的任意一个，那么国王从格子 $A(x_1,y_1)$ 走到格子 $B(x_2,y_2)$ 最少需要多少步？我们会发现最少步数总是 $\max\{|x_2-x_1|,|y_2-y_1|\}$ 步。

对象 i 和 j 的切比雪夫距离定义为

$$d(i,j)=\max\{|x_{i1}-x_{j1}|,|x_{i2}-x_{j2}|,\cdots,|x_{in}-x_{jn}|\} \tag{2-14}$$

闵可夫斯基距离（**Minkowski distance**）是衡量数值点之间距离的一种非常常见的方法，它是欧氏距离与曼哈顿距离的推广，其定义如下：

$$d(i,j)=\sqrt[k]{|x_{i1}-x_{j1}|^k+|x_{i2}-x_{j2}|^k+\cdots+|x_{in}-x_{jn}|^k} \tag{2-15}$$

闵可夫斯基距离不是一种距离，而是一组距离的定义。该距离最常用的 k 是2和1，前者是欧几里得距离，后者是曼哈顿距离。

标准化欧氏距离（**standardized euclidean distance**）是针对简单欧氏距离的缺点而作的一种改进，其基本思想是：先将数据对象的各个分量都进行均值为 μ 或标准差为 s 的标准化，然后再计算欧式距离。其定义如下：

$$d(i,j)=\sqrt{\left(\frac{x_{i1}-x_{j1}}{s_1}\right)^2+\left(\frac{x_{i2}-x_{j2}}{s_2}\right)^2+\cdots+\left(\frac{x_{in}-x_{jn}}{s_n}\right)^2} \tag{2-16}$$

式(2-16)中的 $s_1,s_2,\cdots,s_n$ 分别表示不同属性的标准差，标准化欧氏距离也可以看作是加权的欧氏距离，这种加权思想也可以用于以上其他几种距离度量中。

4. 序值属性相似性

序值属性的值之间具有有意义的序或排位。通过把数值属性的值域划分成有限个类

别，再离散化数值属性得到序值属性。即将数值属性的值域经过离散化映射到具有 M_f 个类别的序值属性。如将身高数值属性离散化到 3 个区间：140～160cm，160～175cm，175～190cm，分别定义这 3 个区间为"矮""中等""高"，于是这些有序的状态定义了一个排位 1，2，…，M_f，数值属性"身高"转换为序值属性"身高"。

对于序值属性刻画的对象，如何来计算其邻近性？假设 f 是用于描述 n 个对象的一组序值属性之一。关于 f 的邻近性计算步骤如下：

（1）第 i 个对象的 f 值为 x_{if}，属性 f 有 M_f 个有序的状态，表示排位 1，2，…，M_f。用对应的排位 $r_{if}\in\{1,2,\cdots,M_f\}$ 取代 x_{if}。

（2）由于每个序值属性都可以有不同的状态数，所以通常需要将每个属性的值域标准化映射到[0，1]上，以便每个属性都有相同的权重。通过用 z_{if} 代替第 i 个对象的 r_{if} 来实现数据标准化，计算公式如下：

$$z_{if}=\frac{r_{if}-1}{M_f-1} \tag{2-17}$$

（3）两两对象之间距离的计算可以用前面介绍的任意一种数值属性的距离度量计算方法，式(2-17)中第 i 个对象的 f 属性值使用 z_{if} 代替。

以上介绍了不同属性类型情况下如何求解对象之间的相似性。然而一个数据分析问题中对象的特征向量并不一定是确定的属性组合，比如寻找文本内容相似的文档，这里相似度的计算主要侧重于判断字符或单词是否重复或完全重复，这时以上所介绍的公式不再适用。下面对相似性计算公式进行扩展，把相似度的概念进一步扩展到集合等运算上。

5. Jaccard 相似性

Jaccard 相似性是一个特定的"相似度"概念，即通过计算两个对象特征集合的交集的相对大小来获得集合之间的相似性。假设两个对象的特征集合分别为 S 和 T，则两个对象的 Jaccard 相似性计算公式如下：

$$\mathrm{sim}(S,T)=\frac{|S\cap T|}{|S\cup T|} \tag{2-18}$$

Jaccard 相似性度量是两个对象特征集合的相似程度，但是它并不是一个真正意义上的距离测度。也就是说，集合越接近，Jaccard 相似性越大，而不是像距离一样应该越短。在许多文献中经常会提到 Jaccard 距离(Jaccard distance)，它是一个表示距离的度量，通过 1 减去 Jaccard 相似性得到。Jaccard 相似性适用于多个应用，包括文档的文本相似度及顾客购物习惯的相似度计算等。

例 2-6　假设有两组数据，$A=\{3,4,5\}$，$B=\{1,2,3,5,6,7\}$。下面计算两组数据的 Jaccard 相似度。

根据式(2-18)，有如下计算过程：

$$\mathrm{sim}(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|\{3,5\}|}{|\{1,2,3,4,5,6,7\}|}=\frac{2}{7}$$

可以得到 A 与 B 的 Jaccard 相似度为 2/7。

6. 编辑距离

编辑距离(edit distance)只适用于字符串比较，其计算方法有两种。一种是字符串 A

到 B 的编辑距离等于将字符串 A 变换为字符串 B 所需要的单字符插入及删除等操作的最小数目。最基本的编辑距离计算中字符操作仅仅包括插入、删除和替换 3 种操作，一些扩展的编辑距离计算方法（如 Damerau - Levenshtein distance 距离）中加入了交换操作。另一种计算编辑距离的方法是基于字符串 A 到 B 的最长公共子序列（Longest Common Subsequence，LCS），通过在 A 和 B 的某些位置上进行删除操作能够构造出它们的最长公共字符串，这时的编辑距离就等于 A 与 B 的长度之和减去它们的 LCS 长度的两倍。

编辑距离应用很广，最初的应用是拼写检查和近似字符串匹配。现在在生物医学领域，科学家将 DNA 看成由 A、S、G、T 构成的字符串，然后采用编辑距离判断不同 DNA 的相似度。另一个很好的用途是在语音识别中，它被当作一个评测指标。语音测试集的每一句话都有一个标准答案，可以利用编辑距离判断识别结果和标准答案之间的不同。

7. 汉明距离

给定一个向量空间，汉明距离（Hamming distance）定义为两个向量中值不相同分量的个数。显然，汉明距离是一种距离测度，其值非负，当且仅当两个向量相等时，汉明距离为 0。其中向量分量的取值可以来自任意集合，但实际使用时常用于布尔向量，即这些向量仅仅包含 0 和 1 的取值。

例 2-7 向量 110110 和向量 101100 的汉明距离为 3，因为这两个向量的第 2、3、5 位元素不同，而其他元素均相同。

8. 余弦相似度

几何中夹角的余弦也可用来衡量两个向量在方向上的差异，在数据分析中经常用余弦相似度（cosine similarity）这一概念来衡量样本向量之间的差异。余弦相似度的取值范围为[−1，1]。其值越大，表示两个向量的夹角越小，两个样本相似性也就越大；其值越小，表示两向量的夹角越大，两个样本相似性也就越小。当两个向量的方向重合时，夹角余弦取最大值 1；当两个向量的方向完全相反时，余弦相似度取最小值−1。

两个 n 维样本向量 $\mathrm{A}(x_{11}, x_{12}, \cdots, x_{1n})$ 和 $\boldsymbol{B}(x_{21}, x_{22}, \cdots, x_{2n})$ 的余弦相似度定义为

$$\mathrm{sim}(\boldsymbol{A}, \boldsymbol{B}) = \frac{\boldsymbol{AB}}{||\boldsymbol{A}|| \times ||\boldsymbol{B}||} \tag{2-19}$$

其中 $||\boldsymbol{A}||$ 是向量 $\boldsymbol{A}$ 的欧几里得范数，定义为

$$||\boldsymbol{A}|| = \sqrt{x_{11}^2 + x_{12}^2 + \cdots + x_{1n}^2} \tag{2-20}$$

也可以说，它就是向量的长度。余弦值为 0 意味着两个向量夹角为 90°，没有匹配。余弦值越接近 1，夹角越小，向量之间的匹配也就越大。

例 2-8 假设有两个词频向量，$\boldsymbol{x}=\{3,0,4,0,1,0,0,6,0\}$，$\boldsymbol{y}=\{1,0,3,0,0,2,0,1,0\}$，可以使用式(2-19)来计算两个向量的余弦相似性。

计算过程如下：

$$\boldsymbol{x} \cdot \boldsymbol{y} = 3\times1+0\times0+4\times3+0\times0+1\times0+0\times2+0\times0+6\times1+0\times0 = 21$$

$$||\boldsymbol{x}|| = \sqrt{3^2+0^2+4^2+0^2+1^2+0^2+0^2+6^2+0^2} \approx 7.87$$

$$||\boldsymbol{y}|| = \sqrt{1^2+0^2+3^2+0^2+0^2+2^2+0^2+1^2+0^2} \approx 3.87$$

$$\text{sim}(\boldsymbol{x},\boldsymbol{y}) \approx 0.69$$

由上述计算可得这两个向量的余弦相似度为0.69,具有较高的相似性。

在以上所介绍的相似性计算公式中,样本所属空间包括欧式空间和非欧空间两种。欧式空间的一个非常重要的性质是空间中点的平均总是存在的,并且也是空间中的一个点,例如欧氏距离的计算。而非欧空间中平均就没有任何意义,比如Jaccard相似性和编辑距离等。余弦相似性的空间可以是欧式空间也可以是非欧式空间,如果向量的分量是任何实数,那么此时就位于欧式空间。但是,如果将向量的分量限定为整数,那么就是非欧空间。

欧氏距离是判断数据对象相似性最常见的距离度量方法,而余弦相似度则是最常见的相似性度量,很多的距离度量和相似性度量都是基于这两者的变形和衍生,所以下面将比较两者在衡量个体差异时实现方式和应用环境上的区别。

从图2-3可以看出,欧氏距离衡量的是空间各点间的绝对距离,和各个点所在的位置坐标(即个体特征维度的数值)直接相关;而余弦相似度衡量的是空间向量的夹角,着重体现在方向上的差异,而不是位置。如果保持A点的位置不变,B点朝原方向远离坐标轴原点,此时余弦相似度$\cos\theta$是保持不变的,因为夹角不变,而A、B两点的距离显然在发生改变,这就是欧氏距离和余弦相似度的不同之处。

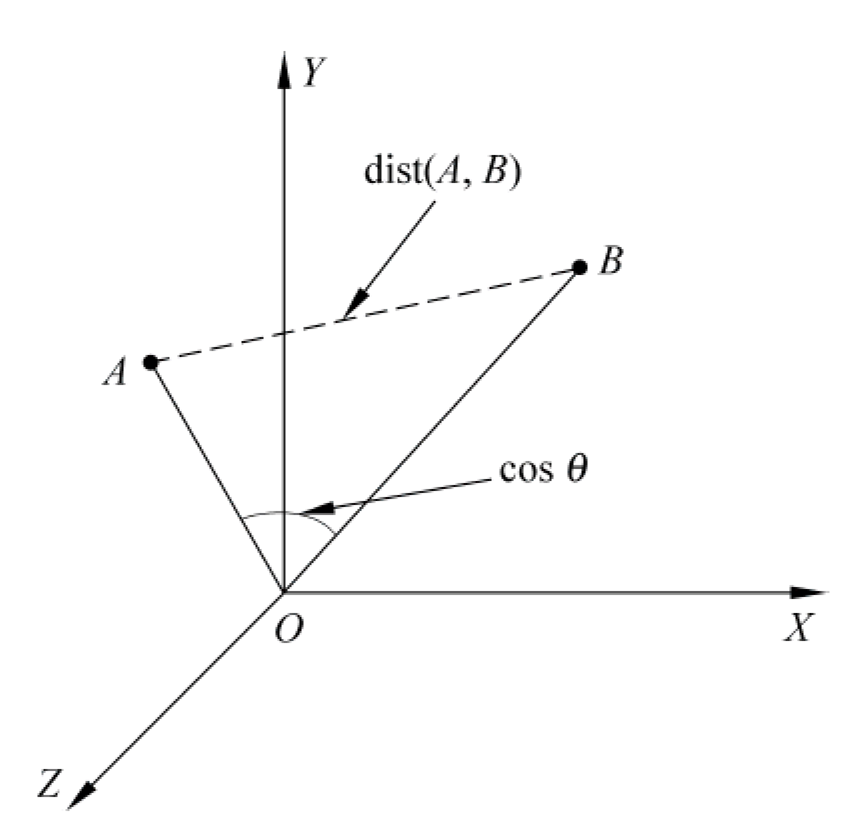


图2-3 余弦相似度与欧氏距离示意图

根据欧氏距离和余弦相似度各自的计算方式和特点,可以看出两者分别适用于不同的数据分析模型:欧氏距离能够体现个体数值特征的绝对差异,所以常用于需要从维度的数值大小中体现差异的分析,如使用用户行为指标分析用户价值的相似度或差异;而余弦相似度侧重于从方向上区分差异,而对绝对的数值不敏感,多用于通过用户对内容评分来区分用户兴趣的相似度和差异,同时修正了用户间可能存在的度量标准不统一的问题。

2.4.2 数据相关性计算方法

世间万物总是存在不同程度的联系。例如身高与年龄、学历与收入或考试成绩与学习时间。对于我们所观察到的数据,同样可能存在着千丝万缕的联系。本节将介绍数据相关性的计算方法。

1. 皮尔逊相关系数

皮尔逊相关系数(Pearson correlation coefficient)也称为简单相关系数,它是一种线性相关系数,是描述两组随机变量X与Y相关程度的一种方法,相关系数的取值范围是$[-1,1]$。相关系数的绝对值越大,则表明X与Y相关度越高。负值表示负相关,即一个

变量的值越大，另一个变量的值反而会越小；正值表示正相关，即一个变量的值越大，另一个变量的值也会越大。要注意的是以上并不代表变量间存在因果关系。若相关系数值为0，表明两个变量间不是线性相关，但有可能是其他方式的相关（比如曲线方式）。其计算公式如下：

$$r(X,Y)=\frac{\sum_{i=1}^{n}(x_i-\overline{X})(y_i-\overline{Y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{X})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{Y})^2}}=\frac{\sum_{i=1}^{n}(x_i-\overline{X})(y_i-\overline{Y})}{\sigma_X\sigma_Y} \tag{2-21}$$

其中 n 为样本个数，$\overline{X}$ 和 $\overline{Y}$ 分别是样本集 X 和 Y 的均值，σ_X 和 σ_Y 是它们的标准差。

2. 斯皮尔曼秩相关系数

斯皮尔曼秩相关系数（Spearman rank correlation coefficient）与皮尔逊相关系数一样，也可以反映两组变量联系的紧密程度，取值在[−1,1]之间，计算方法上也完全相同，不同的是它建立在秩次的基础之上，对原始变量的分布和样本容量的大小不作要求，属于非参数统计方法，适用范围更广。

设 $R(r_1,r_2,\cdots,r_n)$ 表示 X 在 $(x_1,x_2,\cdots,x_n)$ 中的秩，$Q(q_1,q_2,\cdots,q_n)$ 表示 Y 在 $(y_1,y_2,\cdots,y_n)$ 中的秩，如果 X 和 Y 具有同步性，那么 R 和 Q 也会表现出同步性，反之亦然。将其代入皮尔逊相关系数的计算公式，就得到秩之间的一致性，也就是斯皮尔曼秩相关系数。考虑到 $r_1+r_2+\cdots r_n=q_1+q_2+\cdots+q_n=n(n+1)/2$，$r_1^2+r_2^2+\cdots+r_n^2=q_1^2+q_2^2+\cdots+q_n^2=n(n+1)(2n+1)/6$，斯皮尔曼秩相关系数可以定义为

$$r(X,Y)=1-\frac{6[(r_1-q_1)^2+(r_2-q_2)^2+\cdots+(r_n-q_n)^2]}{n(n^2-1)} \tag{2-22}$$

斯皮尔曼秩相关系数对数据条件的要求没有皮尔逊相关系数严格，只要两个变量的观测值是成对的等级评定数据，或者是由连续变量观测数据转化得到的等级数据，不论两个变量的总体分布形态、样本容量的大小如何，都可以用斯皮尔曼秩相关系数来进行研究。

3. 协方差

在概率论和统计学中，协方差用于衡量两个变量的总体误差。而方差是协方差的一种特殊情况，即两个变量相同时的情况。现实生活中，我们经常遇到多维数据，为了了解两个维度数据间是否有一些联系，可以计算其协方差。

期望值分别为 $E(X)=\mu$ 与 $E(Y)=\nu$ 的两个实数随机变量 X 与 Y 之间的协方差定义为

$$\mathrm{cov}(X,Y)=E((X-\mu)(Y-\nu)) \tag{2-23}$$

其中，E 是期望值。上式也可表示为

$$\mathrm{cov}(X,Y)=E(XY)-\mu\nu \tag{2-24}$$

直观上来看，协方差表示的是两个变量总体的误差，这与只表示一个变量误差的方差不同。如果两个变量的变化趋势一致，也就是说，如果其中一个大于自身的期望值，另一

个也大于自身的期望值，那么两个变量之间的协方差就是正值；如果两个变量的变化趋势相反，即其中一个大于自身的期望值，另一个却小于自身的期望值，那么两个变量之间的协方差就是负值。如果 X 与 Y 是统计独立的，那么二者之间的协方差就是 0。但是，反过来并不成立。即如果 X 与 Y 的协方差为 0，二者并不一定是统计独立的。

例 2-9 求两组向量 $\boldsymbol{x}=\{6,4,7,10,8\}$ 与 $\boldsymbol{y}=\{5,6,1,4,12\}$ 的协方差，有如下步骤：

$$E(\boldsymbol{x})=\frac{6+4+7+10+8}{5}=7$$

$$E(\boldsymbol{y})=\frac{5+6+1+4+12}{5}=5.6$$

$$E(\boldsymbol{xy})=\frac{6\times5+4\times6+7\times1+10\times4+8\times12}{5}=39.4$$

$$\mathrm{cov}(\boldsymbol{x},\boldsymbol{y})=E(\boldsymbol{xy})-E(\boldsymbol{x})E(\boldsymbol{y})=39.4-(7\times5.6)=0.2$$

计算得到两组向量协方差为 0.2。

2.5 数据准备

目前比较成熟的数据分析算法，其处理的数据集合都要求数据完整性好、数据冗余少、属性之间的相关性小等条件才能保证分析的质量。然而实际系统中的原始数据往往由于人为疏忽、设备异常或抽样方法等因素，常常会出现数据错误、数据缺失、不一致、重复或矛盾等不同情况，如果直接对这样的原始数据进行分析，很难得到高质量的数据分析结果。另外海量的实际数据中无意义的成分很多，严重影响了数据分析算法的执行效率。通常在获取了原始数据之后，需要对原始数据进行相应的准备工作，其中包括选择数据、清洗数据、数据重构、数据整合、数据归约和变换等。因此，在数据分析之前如何准备好可用数据已经成为数据分析系统实现过程中的关键问题。

数据准备的主要步骤有数据清洗、数据集成、数据归约和数据转换。本节先介绍数据清洗与集成，然后对数据归约技术进行一些探讨，最后讨论关于数据转换的相关知识。

2.5.1 数据清洗与集成

数据清洗(data cleaning)的主要目的是填充或删除遗漏值，降低噪声与识别离群点，纠正数据的不一致。数据集成(data integration)的主要目的是合并来自多个数据存储中的数据，解决多重数据存储或合并时所产生的数据不一致、数据重复或数据冗余问题，有助于提高后续数据分析过程的准确性和速度。以上两种数据准备的步骤有着不同的处理方法，分别阐述如下。

1. 数据清洗

1）缺失值

缺失值为遗漏或错误的数据，可能包括人为或计算机数据输入的错误，输入时理解错误，也有可能是搜集数据的设备出了问题，转换文件时出了问题，造成数据遗失，这些都必须在数据分析前进行清洗，以降低由此对后续数据分析结果的影响。怎样才能为缺失的

属性填上值？以下是几种处理缺失数据的方法：

(1) 直接删除缺失属性的记录。将存在缺失信息的记录删除，从而得到一个完整的数据集。但是这种方法有很大的局限性。它以减少数据量来换取信息的完备，有可能会丢弃大量对数据分析有用的信息。

(2) 人工填写。对于特别小的数据集，这样做当然是有效的。但是当数据量比较大时，这种方法显然是行不通的。

(3) 使用全局常量填充缺失值。把每个缺失的值都用一个全局常量替换，但是这样做很容易干扰分析结果。

(4) 使用属性的中心趋势度量值填充缺失值。中心趋势度量值是指用前面讨论过的计算属性中心趋势的中位数或均值等方法来填充。

(5) 使用与给定元组属于同一类的所有样本的属性均值或中位数填充。

(6) 使用最有可能的值填充。用回归或使用贝叶斯形式化方法的基于推理的工具或决策树归纳确定。

以上第4种方法应用较广，SPSS的Moduler一般直接采用这种方法进行填写，通常对于数值属性用均值替换，标称属性通常用众数替换，序值属性通常用中位数替换。第6种方法也是使用较广的一种方式，因为它使用已有的数据来预测缺失值，因此得出的填充值可信度更高。

2) 识别离群点和平滑噪声数据

噪声数据(noise data)是指测量变量中的随机错误和偏差。它可能会影响数据分析的结果。引起噪声数据的原因有很多，可能是因为数据收集手段、数据输入问题、数据传输问题等。那么，给定一个数值属性，怎么才能"光滑"数据或者去掉噪声？典型的方法有以下几种：

(1) 分箱法(binning)。

分箱法通过考察数据的"近邻"(即周围的值)来光滑有序数据值。这些有序的值被分布到一些箱中，用"箱的深度"表示不同的箱里有数据的个数，用"箱的宽度"来表示每个箱值的取值区间。由于分箱方法考虑相邻的值，因此是一种局部平滑方法。一些常用的分箱方法如下：

- 等宽分箱。将变量的取值范围分为 k 个等宽的区间，每个区间当作一个分箱。
- 等频分箱。把观测值按照从小到大的顺序排列，根据观测的个数等分为 k 部分，每部分当作一个分箱。
- 基于 k 均值聚类的分箱。使用第6章将介绍的 k 均值聚类法将观测值聚为 k 类，但在聚类过程中需要保证分箱的有序性，即第一个分箱中所有观测值都要小于第二个分箱中的观测值，第二个分箱中所有观测值都要小于第三个分箱中的观测值，等等。

采用上述方法中的一种之后，可以使用箱均值光滑、箱中位数光滑或箱边界光滑的方法来平滑噪声，用一个例子可以很容易地理解。

例 2-10 假设有一组观测值：8，9，30，24，10，15，24，7，28，从小到大排序得：7，8，9，10，15，24，24，28，30。采用等频分箱将该组值分为3箱：箱1(7，8，9)，箱2(10，15，24)，

箱 3(24,28,30)。

- **用箱平均值光滑**。对于箱 1,其平均值为 8。将箱 1 中的所有值替换为 8,即,箱 1(8,8,8)。
- **用箱中位数光滑**。对于箱 2,其中位数为 15。将箱 2 中的所有值替换为 15,即,箱 2(15,15,15)。
- **用箱边界光滑**。对于箱 3,箱中的最大值和最小值分别为 30 和 24,箱中每一个值都被最近的边界值替换,即,箱 3(24,24,30)。

一般来说,箱的区间范围越大,光滑效果越好。

(2) 回归(Regression)。

用一个函数拟合数据来光滑数据,这种技术称为回归。回归分析中,只包括一个自变量和一个因变量,且二者的关系可用一条直线近似表示,这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量,且因变量和自变量之间是线性关系,则称为多元线性回归分析。

例 2-11 有下面一组数据:

y=[20 20 21 22 22 22 23 24 26 28 56 34 43 45 45.37 48.07 54.07 54.08 56.81 59.45]

x=[14.81 15.34 16.13 17.6 18.59 20.3 21.52 22.49 24.2 25.22 26.56 27.76 27.79 44.35 45.61 44.52 45.57 46.98 46.14 48.36]

用 MATLAB 进行线性拟合,图 2-4 中的直线即为一条拟合线。

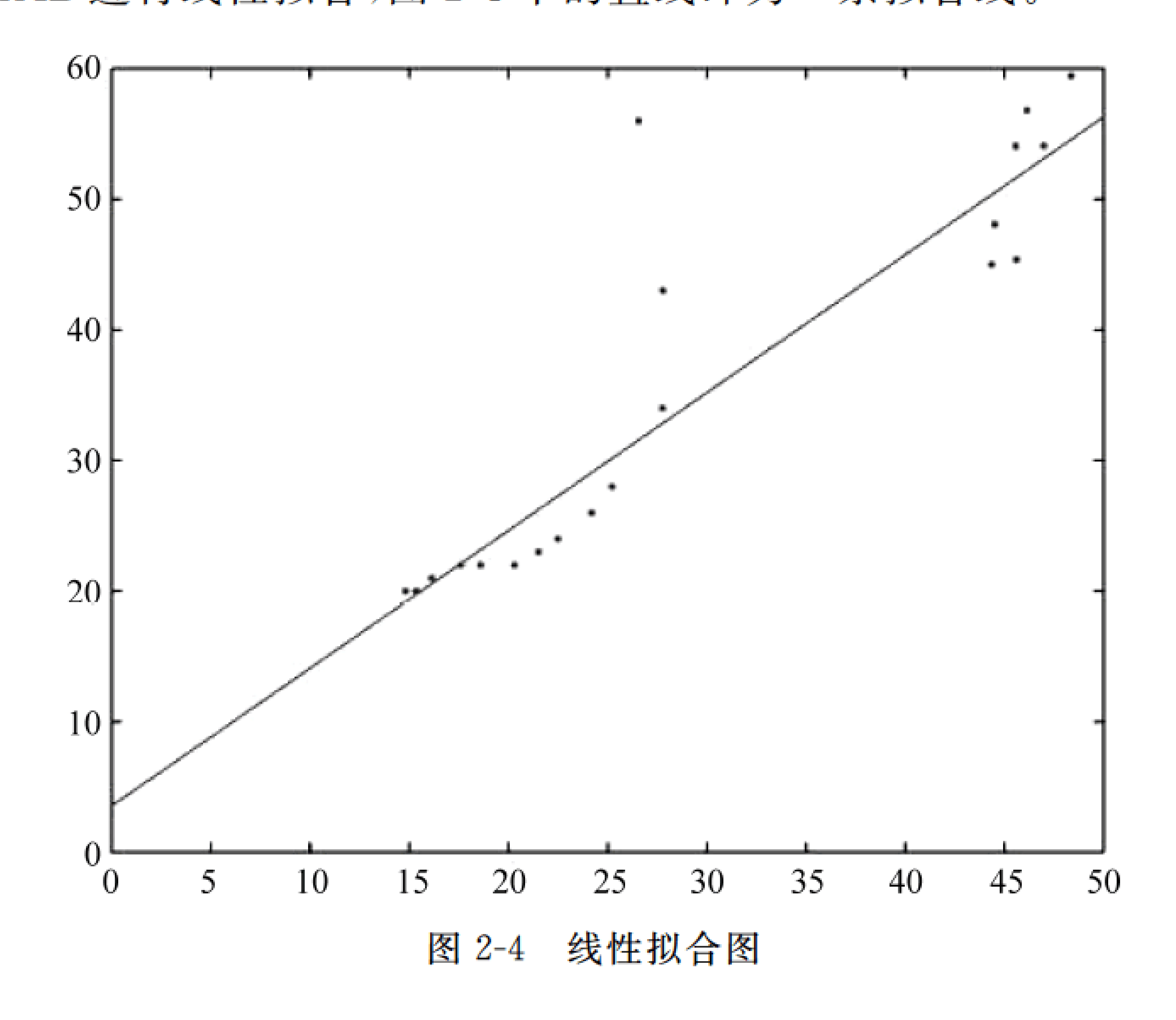


图 2-4 线性拟合图

(3) 离群点分析(Outlier analysis)。

在样本空间中,与其他样本点的一般行为或特征不一致的点称为离群点。可以通过像聚类的方法来检测离群点,聚类可以将相似点聚成簇,而落在簇外的点被视为离群点。离群点的产生可能是计算的误差或者操作的错误引起的,也可能是因为数据本身的可变

性或弹性所致。

例 2-12 对于例 2-11 的数据,还可以找出它的离群点,在图 2-5 中可以看到,在两个类簇之外有 3 个孤立点,这 3 个点就是离群点。

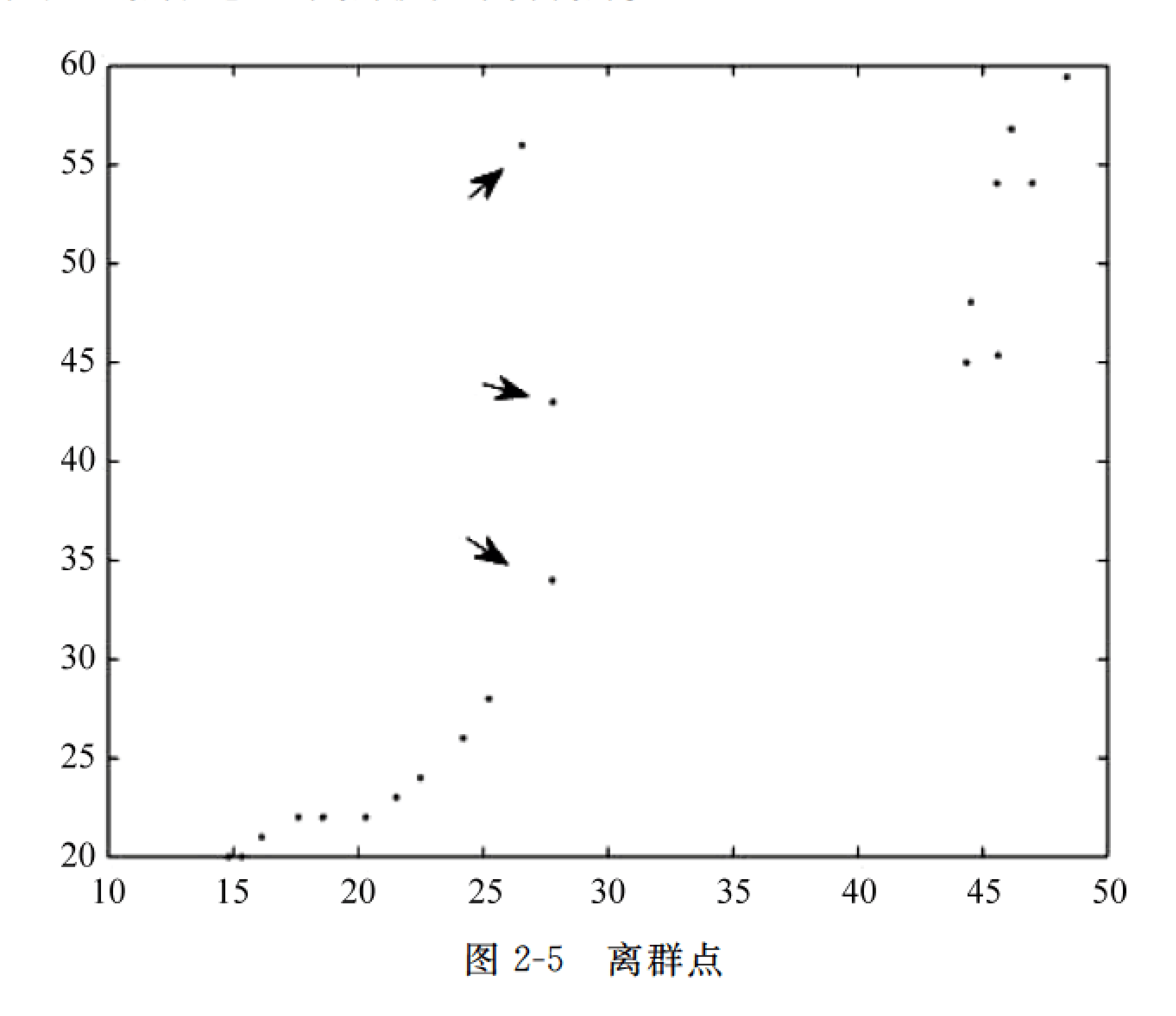


图 2-5 离群点

2. 数据集成

1) 异常数据

当数据分析的数据集来自不同数据源时,会出现许多不同类型的数据异常情况,因此保证所分析数据的一致性是必须考虑的问题。这种不一致性主要表现在数据对象名称与同类型其他数据对象名称不相符、数据对象的特征与其他同类对象不相符以及数据对象特征的取值范围不一致等,这些都属于实体识别问题。如何才能确保来自多个数据源的实体"匹配"? 首要方法是分析各个数据源中的元数据,每个数据对象以及它们特征的元数据包括名称、含义、数据类型、属性的取值范围以及空值规则。这些元数据可以用来帮助避免数据集成产生的错误。

2) 冗余数据

冗余是数据集成期间可能遇到的另一个重要问题。一个属性如果可以由另一个或另一组属性导出,则这个属性可能就是冗余的。属性的冗余容易造成数据量过大、数据分析时间过长、结果不稳定的问题,因此如何判别冗余属性也是数据集成中的一个重要步骤。通常会采用相关性分析手段来检测冗余。例如给定两个属性,采用前面介绍过的某种相关性判别方法度量一个属性能在多大程度上蕴含或依赖另一个属性,这样就很容易鉴别出哪个是冗余属性了。

3) 重复数据

除了检测对象特征间的冗余外,还应检测对象的重复。重复是指对于同一数据集,存在两个或多个相同的数据对象,或者相似度大于阈值的数据对象。这通常都是由于不正

确的数据输入，或者更新了数据后，以前陈旧的数据没有删除等导致的。

数据清洗和数据整理是解决数据不一致性的两种重要方法，数据的不一致可能导致数据分析结果的偏差。除了采用上面介绍的各种方法进行人工纠错外，目前已经有许多商业工具可以帮助我们检测不一致性并同时予以纠错。数据清洗工具（data scrubbing tool）使用简单的领域知识检查并纠正数据中的错误，它们通常采用语法分析和模糊匹配技术完成对多数据源数据的清理。常见的数据清洗工具有：①DataWrangler，是基于网络的服务，是斯坦福大学的可视化组设计的，用来清洗和重排数据；②佳数 rightdata，是国内第一个以 SAAS 模式提供完整地址数据处理服务流程的网站；③OpenRefine，前身是 Google Refine，它是一个开源的用于进行数据清理的工具，其主要功能是清洗数据以及数据转换等。数据审计工具（data auditing tool）可以通过扫描数据发现规律和联系，并检测违反这些条件的数据来发现偏差，类似这样的数据清洗和数据审计工具在很多商用的统计分析工具中都提供了，可以根据使用过程中具体情况加以选择。

2.5.2 数据归约

有时候，得到的数据集非常庞大，在这样的数据集上进行复杂的数据分析是非常耗时的。这时就要考虑能否得到这个数据集的一个简化版本，但同时又不会影响数据分析的结果。数据归约（Data reduction）技术有助于得到简化的数据集，它小得多，但是仍能相对地保持数据的完整性，在其上进行数据分析能产生与在原数据集上进行分析几乎相同的结果。

数据归约技术包括维归约、数量归约和数据压缩。

维归约（dimensionality reduction）减少样本空间中所包含的属性个数。其方法包括小波变换、主成分分析以及属性子集选择，前面两种是把原数据变换或投影到维数较小的样本空间中，而后者是通过相关性等方法分析后，检测样本空间中不相关、弱相关或冗余的属性或维，然后予以删除。

数量归约（numerosity reduction）用替代的、较小的数据集表示形式替换原数据集，包括参数方法或非参数方法。参数方法就是为数据集拟合一个描述模型来估计数据，使得存储形式只需要存放模型参数，而不是实际的数据集，例如各种回归模型。非参数方法包括直方图、聚类和抽样等。

数据压缩（data compression）使用不同变换方法以得到原数据的压缩形式。如果原数据能够从压缩后的数据重构，而不损失信息，则该数据归约技术称为无损压缩。无损压缩用于要求重构的信号与原始信号完全一致的场合。一些常用的无损压缩算法有赫夫曼（Huffman）算法和 LZW（Lenpel-Ziv & Welch）压缩算法。如果只能近似重构或不完全恢复原数据，则该数据压缩技术称为有损压缩。有损压缩广泛应用于语音、图像和视频数据的压缩。它利用了人类对图像或声波中的某些频率成分不敏感的特性，允许压缩过程中损失一定的信息，虽然不能完全恢复原始数据，但是所损失的部分对理解原始图像的影响很小，却换来了大得多的压缩比。由于数据压缩方法涉及许多压缩算法，本节不做介绍。

1. 维归约

1）小波变换

离散小波变换(Discrete Wavelet Transform，DWT)是一种线性信号处理技术，当用于数据向量 X 时，就是将它变换成数值上不同的小波系数向量 X'，两个向量具有相同的长度。当把这种技术用于数据归约时，每个样本可以看作一个 n 维数据向量，即 $X=(x_1,x_2,\cdots,x_n)$，该向量描述了数据集数据样本在 n 个属性上的测量值。经过小波变换后得到的数据向量长度与原始数据向量的长度相等，但是其中一半是代表属性的，另一半是对属性的描述，称为小波系数。可以通过截断小波变换后的数据向量，保留大于用户设定的某个阈值的所有小波系数而其他系数置为 0 的方式来获得近似的压缩数据。这样，结果数据向量表示长度可以减小，使得在小波空间进行计算时速度能够加快。该技术也能用于消除噪声，而不会光滑掉数据的主要特征，可以有效地用于数据清理。对于变换后得到的一组系数，可以使用所用的小波变换的逆来构造原数据的近似，所以小波变换属于有损压缩。

DWT 与离散傅里叶变换(Discrete Fourier Transform，DFT)有密切关系，DFT 是一种只涉及正弦和余弦的信号处理技术。相对于 DFT 来说，DWT 是一种更好的有损压缩方式。对于给定的数据向量，如果 DWT 和 DFT 保留相同数目的系数，DWT 将提供更准确的原数据近似。而且 DWT 比 DFT 需要的空间更小，局部细节保留的效果更好，对于稀疏或倾斜数据以及具有有序属性的数据，可以得到更好的结果。

流行的小波变换包括 Haar-2、Daubechies-4 和 Daubechies-6 变换，图 2-6 是在 MATLAB 平台上对 noissin 数据集进行 haar 小波变换后的图像，其中 ca 和 cd 分别为低频系数向量和高频系数向量。

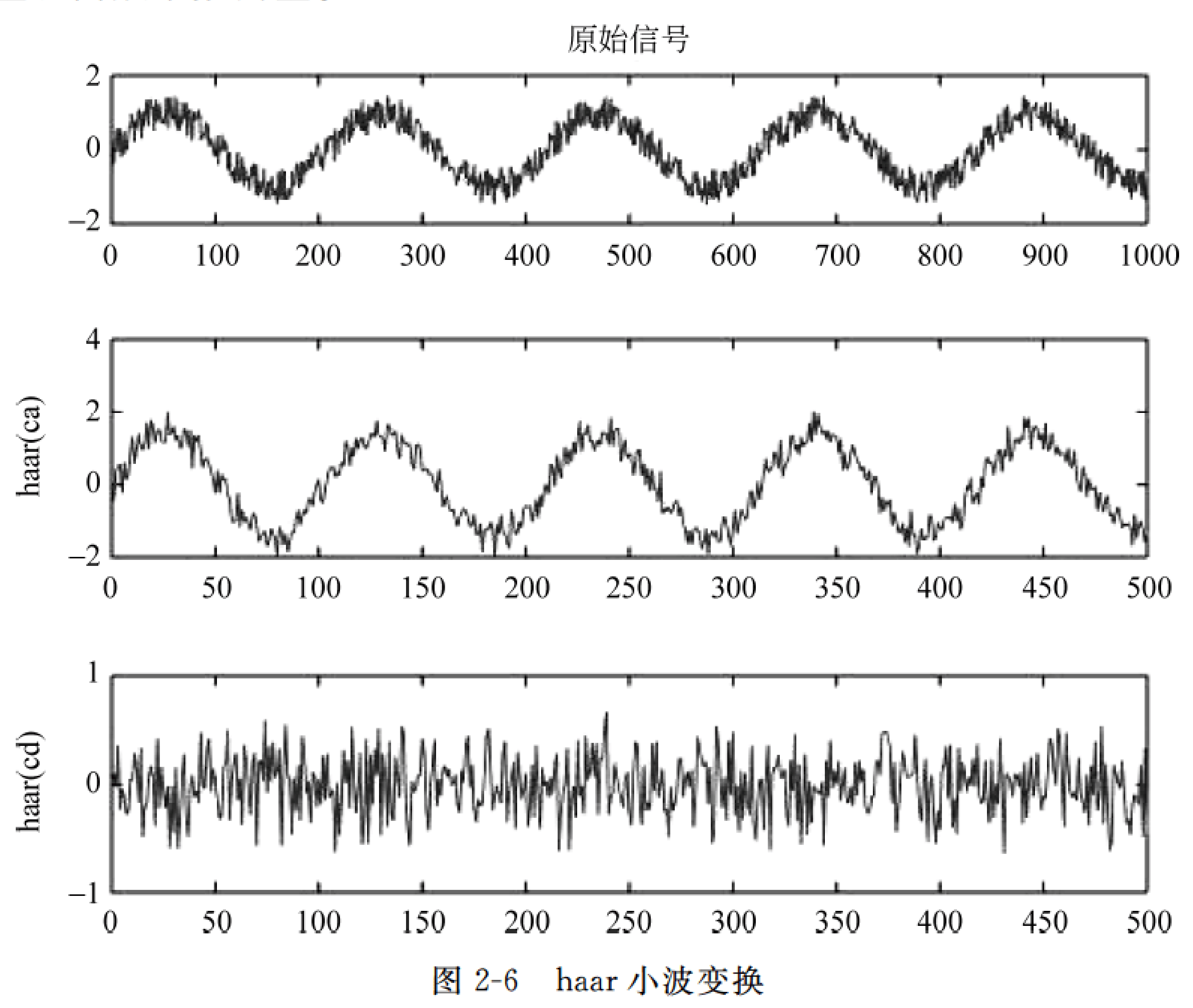


图 2-6 haar 小波变换

应用离散小波变换的算法过程称为分层金字塔算法(pyramid algorithm),离散小波变换的一般步骤如下:

(1) 输入数据向量的长度 L 必须是 2 的整数幂(必要时,可以在数据向量后加 0)。

(2) 每个变换涉及两个应用函数:平滑和差分,前者用于光滑数据,经常使用求和或加权,后者用于提取数据的细节特征。

(3) 两个函数作用于输入的数据向量,产生两个长度为 $L/2$ 的数据集。

(4) 两个函数递归地作用于前面循环得到的数据集,直到结果数据集的长度为 2。

(5) 由以上迭代得到的数据集中选择满足阈值的值作为数据变换后的小波系数。

例 2-13 假设有一组数据{8,6,3,5},对其进行 haar 小波变换,步骤如下:

(1) 求均值。计算相邻数据对的平均值,结果为{7,4}。

(2) 求差值。很明显,用两个数据表示所有数据时,已经丢失了数据部分信息。为了能重构出所有数据,需要存储一些细节系数。将数据对的第一个数值减去这对数值的平均值可得第一个细节系数为 1,第二个细节系数为−1。此时可以用两个平均值与两个细节系数表示原始数据,即{7,4,1,−1}。

(3) 重复第(1)、(2)步,将上面分解得到的数据进一步分解为{5.5,1.5,1,−1}。

通过上述步骤就将由 4 个数据点组成的数据集用一个平均数据值和三个细节系数表示出来了。

2) 主成分分析

真实的数据集总是存在各种各样的问题,比如噪声或者冗余。假设拿到一个数学系的本科生期末考试成绩单,里面有 3 列,分别是对数学的兴趣程度、复习时间和考试成绩。显然要学好数学,需要有浓厚的兴趣,所以第二列与第一列强相关,第三列和第二列也是强相关。那是不是可以合并第一列和第二列呢?再比如,拿到一个数据集,样本特征非常多,而样本个数特别少。对这样的数据集做数据分析时,容易产生过度拟合问题。这时需要一种特征降维的方法来减少特征数,降低噪音和冗余,减少过度拟合的可能性。主成分分析(Principal Components Analysis,PCA)的方法可以解决上述问题。PCA 的思想是将原来数据集中的 n 维特征映射到 k 维上($k<n$),这 k 维是全新的正交特征,称为主成分,它们是重新构造出来的 k 维特征,而不是简单地从 n 维特征中去除其余 $n-k$ 维特征。主成分分析通过搜索 k 个最能代表数据的 k 维正交向量($k\leqslant n$)将原数据投影到一个小得多的空间上,导致维归约。由此,原数据在投影到这个较小的向量空间后,每一个数据向量变换为 k 个主要成分向量的线性组合。

主成分分析基本过程如下:

(1) 对输入数据规范化,使得每个属性都落入相同的区间。

(2) 计算 k 个标准正交向量,作为规范化输入数据的基,这些向量称为主成分。

(3) 对主成分按"重要性"(这里的重要性是按照数据的方差排序)或强度降序排列。

(4) 去掉较弱的主成分,仅仅使用最强的主成分,重构原数据的近似。

与小波变换相比,主成分分析能够更好地处理稀疏数据,而小波变换更适合高维数据。

例 2-14 假设研究某一问题涉及两个维度:A 与 B,其中 A 单位为百万,B 单位为

万。得到的原始数据矩阵如下：

$$\boldsymbol{X}=\begin{bmatrix}12.5 & 586\\ 24 & 754\\ 15.3 & 850\\ 18 & 667\\ 31.2 & 750\end{bmatrix}$$

计算原始数据的协方差矩阵与相关矩阵分别为

$$\boldsymbol{\Sigma}=\begin{bmatrix}55.90 & 242.65\\ 242.65 & 9927.80\end{bmatrix}$$

$$\boldsymbol{R}=\begin{bmatrix}1 & 0.3257\\ 0.3257 & 1\end{bmatrix}$$

由协方差矩阵求解主成分，得到结果见表2-6。

对应两个特征值的标准正交特征向量见表2-7。

表2-6 主成分结果

特征值	解释方差比例	累积比例
9933.7607	0.9950	0.9950
49.9343	0.0050	1.0000

表2-7 特征向量

特征值1	特征值2
0.0246	0.9997
0.9997	−0.0246

因此，所得的主成分的表达式为

$$Y_1=0.0246(X_1-\overline{X_1})+0.9997(X_2-\overline{X_2})$$

$$Y_2=0.9997(X_1-\overline{X_1})-0.0246(X_2-\overline{X_2})$$

其中，第一主成分保留了原始变量99.5%的信息，在分析中就可以把第二主成分舍掉，这样达到简化问题的目的。第一主成分与原始变量的因子负荷量分别为

$$\rho(Y_1,X_1)=\gamma_{11}\frac{\sqrt{\lambda_1}}{\sqrt{\sigma_{11}}}=0.0246\times\frac{\sqrt{9933.7607}}{\sqrt{55.9}}=0.3297$$

$$\rho(Y_1,X_2)=\gamma_{12}\frac{\sqrt{\lambda_1}}{\sqrt{\sigma_{22}}}=0.9997\times\frac{\sqrt{9933.7607}}{\sqrt{9927.80}}=0.9871$$

由此可知，第一主成分反映了数据集98.71%的信息，方差较大的维度对第一主成分起了主要作用。

3）属性子集选择

有时候，要处理的数据集可能包含非常多的属性，而这些属性中只有很少一部分是和当前数据分析任务有关的。这时，可以人工选择需要的属性，然而大部分情况下，这样做是不现实的，因为有时候可能并不能确定哪些属性有用。而过多冗余的属性意味着更多冗余的数据和计算量的增大，这也会降低数据分析的效率。

属性子集选择可以解决上述问题，它通过删除不相关或冗余的属性找出最小属性集，

使得数据集中类的分布尽可能地接近使用所有属性得到的类分布。在子集选择中,需要选择最佳子集,其包含的维度最少,但对正确率的贡献最大。在维度较大时,可以采用启发式方法,在合理的时间内得到一个合理解(但不是最优解);维度较小时,对所有子集做穷举检验。

对于 n 个属性,有 2^n 个可能的子集,通过穷举出这些属性的子集找到最佳子集可能是不现实的。在实践中,一些基于贪心策略的压缩搜索空间的启发式算法往往能得到较好的结果。其基本方法包括逐步向前选择、逐步向后删除、逐步向前选择和逐步向后删除组合以及决策树归纳。

(1) 逐步向前选择。即从空集开始逐渐增加特征,每次添加一个降低误差最多的变量,直到进一步添加后再不会降低误差或者降低很少为止。同时可以用浮动搜索,每一步可以改变增加和去掉的特征数量,以此来加速。

(2) 逐步向后删除。从所有变量开始,逐个排除它们,每次排除一个降低误差最多的变量,直到进一步的排除会显著提高误差。如果预测估计存在许多无用特征时,逐步向前选择更可取。

(3) 逐步向前选择和逐步向后删除组合。可以将逐步向前选择和逐步向后删除方法结合在一起,每一步选择一个最好的属性,并在剩余属性中删除一个最差的属性。

(4) 决策树归纳。决策树算法最初是用于分类的,这些方法将在第 5 章进行讨论。决策树归纳最后能够构造一个类似于流程图的结构,其中每个内部(非叶)节点表示一个属性上的测试,每个分枝对应于测试的一个结果,每个外部(叶)节点表示一个类预测。在每个节点上,算法选择“最好”的属性,将数据集划分成不同子集。当决策树归纳用于属性子集选择时,由给定的数据构造决策树,出现在树中的属性形成归纳后的属性子集。

在以上的属性选择方法中,进行误差检测时应在不同于训练集的测试集上做,因为需要得到检验的泛化准确率。通常使用的特征越多,一般训练误差会越低,但不一定有更低的验证误差。属性子集选择在有些类型的数据分析中不能用于进行求属性子集的操作。比如图像处理中,因为个体像素本身并不携带很多识别信息,能够携带图像识别信息的通常都是许多像素值的组合。

另外为了能够发现数据属性间联系的缺失信息,有助于新的知识发现和数据分析,可以基于其他属性进行分析,构造一些新属性,这称为属性构造,它有助于提高准确性和对高维数据结构的理解。

2. 数量归约

1) 回归分析

回归分析(regression analysis)是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法,运用十分广泛。从不同角度对回归分析进行划分,可以分成很多种类。例如,按照涉及变量的多少,分为一元回归和多元回归分析;按照自变量和因变量之间的关系类型,分为线性回归分析和非线性回归分析;按照因变量的多少,分为简单回归分析和多重回归分析。如果在回归分析模型中只包括一个自变量和一个因变量,且二者

的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析；如果包括两个或两个以上的自变量，且自变量之间存在线性相关性，则称为多元线性回归分析。

一般来说，回归分析的实质是通过规定因变量和自变量来确定变量之间的因果关系，建立回归模型，并根据实测数据来求解模型的各个参数，然后评价回归模型是否能够很好地拟合实测数据。如果能够很好地拟合，则可以根据自变量作进一步预测。同样可以通过为数据集拟合回归模型给数据集建模，于是数据集就可以被压缩成只需要存储对应模型中自变量的属性值和回归模型参数，而由模型可以求得的因变量的属性值不再需要存储，从而大大降低数据的存储量。

还有一种特殊的回归模型是对数线性模型(log-linear model)，它可以描述近似离散的多维概率分布。对于 n 维样本的数据集合，可以把每个样本看作 n 维空间中离散的点，采用对数线性模型，生成基于维组合的一个较小子集，估计多维空间中每个点的概率，从而使数据集从高维数据空间降到低维数据空间。

回归分析不仅可以用于处理稀疏数据，而且可以处理倾斜数据，其中线性回归模型表现最佳。但是对于高维数据而言，线性回归模型计算量很大，而对数线性模型表现性能较好，可以处理10维左右的数据集。

2）直方图

直方图对数量的归约采用的是一种类似于分箱的技术来约简数据的规模，是一种流行的简单数量归约形式。这种方法是将属性按照取值范围划分为不相交的连续区间，每个区间对应一个桶。可以用一个桶代表属性的一个取值，则该桶称为单值桶。通常，桶和属性值的划分可以有两种规则：等宽和等频。等宽直方图的每个桶宽度是一样的，而在等频直方图中，每个桶的频率大致为常数，也就是说，每个桶包含的邻近数据样本个数大致相同。

对于近似稀疏和稠密数据以及高倾斜和均匀的数据，直方图都是非常有效的。单值属性直方图也可以推广到多个属性，称为多维直方图(如例2-15)，由此可以看出多维直方图能够描述属性间的依赖关系。

例 2-15 假设有一个包含3个维度的数据集，各个数据对象三维的取值分别为{1,2,1,3,3,5,3,2,2,1}，{1,2,1,3,3,5,3,2,2,1}，{2,4,2,6,6,10,6,4,4,2}，使用MATLAB绘制该数据集的多维直方图，如图2-7所示。显然从图中可以直观地看出该数据集的3个维度之间存在明显的倍数关系。

3）聚类

数据分析中的聚类技术也能够用于数量归约。它首先把数据样本看做对象，进而将对象划分为簇，使得一个簇中的对象“相似”，而不同簇之间的对象“相异”。在数量归约中，一个簇中的所有数据样本可以用这个簇名进行替换。如果数据集中的数据具有聚集效应，聚类技术非常有效；但是当数据集中噪声比较多时，聚类技术将失去有效性。第6章中将介绍一些经典常用的聚类算法。

4）抽样

除了上述方法，还可以用抽样方法进行数量归约。抽样能够使得大型数据集用小得多的随机数据子集来替代。常用的抽样方法包括：

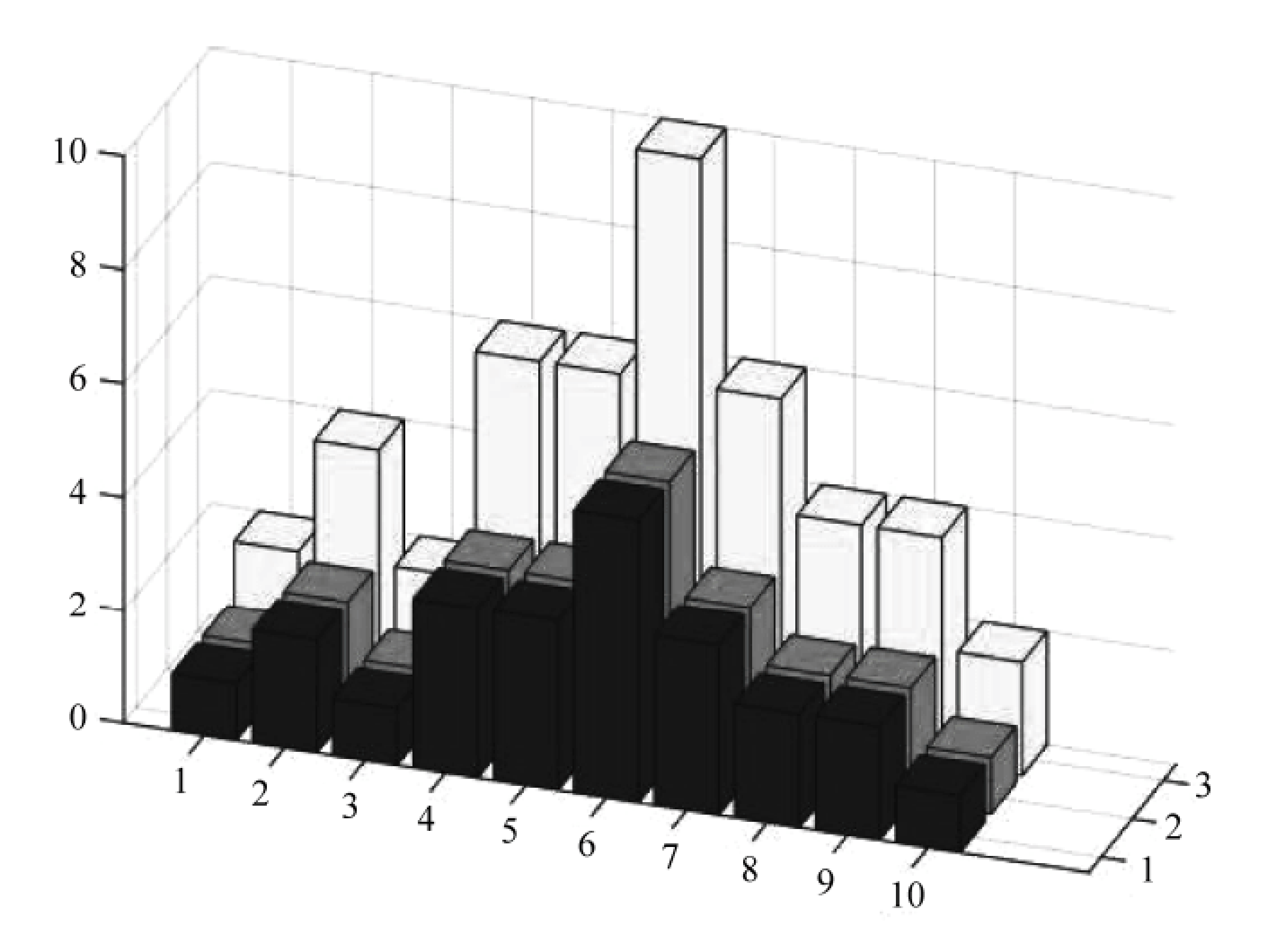


图 2-7　多维直方图

(1) 简单随机抽样。假设一个数据集中样本个数为 N，如果每个样本被抽到的概率相等，且每次抽取时只抽取一个样本，这样的抽样方法称为简单随机抽样。简单随机抽样又分为无放回简单随机抽样和有放回简单随机抽样。简单随机抽样适用于数据集较小的情况。

(2) 分层抽样。抽样前将数据集分成互不相交的层，然后按照一定的比例从各层中独立抽取一定数量的样本，这样的抽样方法为分层抽样。适用于由差异明显的不同部分组成的数据集。

(3) 聚类抽样。是将数据集中各单位归并成若干个互不交叉、互不重复的集合，称为簇，然后以簇为抽样单位进行抽样。应用聚类抽样时，要求各簇有较好的代表性，即簇内各样本的差异要小，簇间差异要大。

采用抽样进行数据分析，计算量正比于样本集的大小，与数据集大小无关。而其他数量归约技术都需要完整地扫描数据集。因此采用抽样进行数量归约的数据分析复杂度仅随数据集的维数 n 线性地增加，而其他类似直方图的数量归约技术的复杂度会随维数 n 呈指数增长。

2.5.3　数据转换

在数据准备阶段对数据进行转换有可能使得分析过程更有效，得到的模式更容易理解。这里需要说明的是，数据准备中的数据清洗与集成、数据归约以及数据转换的各个步骤并不是严格划分的，这些阶段可能存在很多重叠，前两种在前面就已经介绍过了，剩下的数据转换过程，其策略主要包括规范化和离散化等。本节将着重介绍数据转换的相关方法。

1. 数据规范化

数据规范化有时又被称为数据标准化。数据集中数据对象总是用多个属性特征进行

描述，不同属性常常具有不同的单位和不同的取值范围。因此在求取对象间距离时，不同取值单位的属性放在一起进行运算常使解释发生困难。例如，第1个属性的单位是kg，第2个属性的单位是cm，那么在计算绝对距离时将出现两个对象中第1个属性之差的绝对值（单位是kg）与第2个属性之差的绝对值（单位是cm）相加的情况，也就是造成5kg的差异和3cm的差异相加的问题。另外，不同属性的取值范围也可能相差较大，以至于造成在计算对象间绝对距离时某个属性所占的比重太大，对结果的影响也会很大，比如第1个属性的取值范围在2%～4%之间，而第2个属性的取值范围都在1000～5000之间。为了避免单位不同和取值范围不同的影响，数据应该规范化或标准化，使之落入较小的无量纲的共同区间，比如[－1.0,1.0]或[0.0,1.0]。

数据的规范化可有效地改进涉及距离度量的算法的精度和有效性。对于采用距离度量的分类算法和聚类算法，数据规范化是特别重要的步骤。例如，使用神经网络后向传播算法进行分类挖掘，对于训练样本中每个属性的输入值规范化将有助于加快学习阶段的速度；对于基于距离的方法，规范化可以帮助防止具有较大值域的属性相比具有较小值域的属性权重过大。数据规范化的方法有许多，这里着重讨论其中的3种方法：最小-最大规范化、z-score规范化和按小数定标规范化。

1）最小-最大规范化

这种方法对原始数据进行线性变换。假定 $\min_A$ 和 $\max_A$ 分别为属性 A 的最小值和最大值。最小-最大规范化通过计算

$$v' = \frac{v - \min_A}{\max_A - \min_A}(\text{new_}\max_A - \text{new_}\min_A) + \text{new_}\min_A \tag{2-25}$$

将 A 的值 v 映射到区间[new_min_A, new_max_A]中的 v'。

最小-最大规范化能够保持原始数据值之间的联系，但是如果数据集中新增的数据在 A 属性上的值超出了 A 的原始数据值域，该方法将无效。最小-最大规范化方法容易受到离群点的影响，因为离群点很可能是属性 A 取值区间的最大或最小值。

例 2-16 2.7.6节中，数据集包含5个维度，其中salary和salbegin两个维度的取值范围相对于其他维都太大了，在计算数据间距离之前，需要将维度标准化。若salary的最大值为135 000元，最小值为15 750元。对于id为1的实体，其salary值为57 000元，采用最小-最大规范化方法计算如下。

由式(2-26)计算

$$\frac{57\ 000 - 15\ 750}{135\ 000 - 15\ 750} \times (1-0) + 0 \approx 0.35$$

则该实体salary属性规范后的结果为0.35。

2）z分数（z-score）规范化（或零均值规范化）

这种方法基于属性的均值和标准差来进行规范化。其计算公式如下：

$$v' = \frac{v - \overline{A}}{\sigma_A} \tag{2-26}$$

其中 $\overline{A}$ 表示属性 A 所有取值的均值，σ_A 表示属性 A 所有取值的标准差。当不知道属性 A 的最大值和最小值，或者为避免离群点的影响时，可以用z-score规范化。

例 2-17 同样对于例2-16中的salary变量，其均值为34 349元，标准差为16 928

元，对于 id 为 1 的观测值 57 000 元，可以用 z-score 方法对其进行规范化：

$$\frac{57\ 000-34\ 349}{16\ 928}\approx 1.34$$

得到规范化后的结果为 1.34。

3）小数定标规范化

这种方法通过移动属性值的小数点位置进行规范化。小数点的移动位数依赖于属性 A 所有观测值中的最大绝对值。其计算公式如下：

$$v_i'=\frac{v_i}{10^j} \tag{2-27}$$

其中，j 是使得 $\max(|v_i'|)<1$ 的最小整数。

例 2-18 对于例 2-16 中的 salary 属性，因为属性最大绝对值为 135 000 元，所以在小数定标规范化时，将每个属性值除以 1 000 000（即 $j=6$）。对 id 为 1 实体的属性 salary 进行小数定标规范化，由式(2-27)计算

$$\frac{57\ 000}{10^6}=0.057$$

得到规范化后的值为 0.057。

2. 数据离散化

数据离散化是指把连续型数据切分为若干“段”，也称 bin，是数据分析中常用的手段。在营销数据的分析挖掘中，离散化应用非常普遍。其主要原因表现为以下两点：

（1）算法输入数据的需要。例如决策树、Naive Bayes 等算法本身不能直接使用连续型数据，只有经过离散化处理之后才能处理，这一点在使用商用统计分析软件时可能不明显。因为大多数此类软件在各个算法使用时已经内建了离散化处理程序，所以从使用界面看，软件可以接纳任何形式的数据。但实际上，软件都要在后台对连续型数据做预处理。

（2）消除极端数据的影响。离散化可以有效地克服数据中隐藏的缺陷，使获得的分析模型更加稳定。例如，数据中的离群点或噪声是影响分析模型的一个重要因素，它们会导致模型的参数过高或过低，而离散化，尤其是等距离散化，可以有效地降低离群点和噪声的影响。

对数据进行离散化切分的原则有等距、等频、聚类，或根据数据特点等而定。其中等距和等频离散化类似于前面介绍的噪声光滑技术中的分箱法。

（1）等距离散化。将连续型变量的取值范围均匀划成 n 等份，每份的间距相等。例如，在电信行业客户流失分析中，客户入网时间是一个连续型变量，可以从几天到几年。可以采取等距切分的方法把 1 年以下的客户划分成一组，1～2 年的客户为一组，2～3 年为一组……以此类推，间距为一年。

（2）等频离散化。将观察点均匀分为 n 等份，每份内包含的观察点数相同。设该电信公司共有 5 万用户，等频离散化首先需要按照用户的入网时间顺序排列，排列好后可以按 5000 人一组，把全部用户均匀分为 10 份。

等距和等频离散化在大多数情况下会导致不同的结果。其中等距离散化可以保持数

据原有的分布,分箱数越多,对数据原貌就保持得越好。但是当存在对于划分区间而言偏斜极为严重的离群点时,这种离散化方法是不适用的。等频离散化则把数据变换成均匀分布,虽然其避免了等距离散化的问题,但其可能将观察值相同的数据样本分在不同的箱内。不论是等距还是等频离散化,尽管使用简单,易于操作,但是都需要人为定义所划分的区间个数,因此使用时对区间个数的多少极为敏感。

(3) 聚类离散化。该方法包括两个步骤,首先将连续属性的值用聚类算法划分成簇,然后将聚类得到的簇再进行处理,可分为自顶向下的分裂或自底向上的合并策略。

(4) 有监督的离散化。使用上面介绍的3种离散化方法通常会比不做离散化效果要好,但是如果使用附加的信息(比如类标号)进行有监督的离散化,常常能够产生更好的结果。因为未使用类标号信息所构造的区间常常存在不同类对象的混合。有监督的离散化通常使用的最简单方法是以极大化区间纯度来确定分割点,一些基于统计学的方法常被用于为每个属性值分隔区间,常用的划分方法有卡方、信息增益、基尼指数等。

虽然已有了很多离散化方法,但是没有任何一种离散化方法具有普适性,也不存在哪一种离散化方法得到的结果一定好于其他离散化方法,因为离散化本身就是一个NP难问题,因此使用时一定要根据数据集的特点使用合适的离散化方法,以取得尽可能好的离散化效果。

2.6 数据统计分析常用工具介绍

“工欲善其事,必先利其器。”对于数据分析相关工作人员来说,一个趁手的工具可以节省很多时间,大大提高工作效率。对于学生来说,掌握几种流行的统计分析工具是大有益处的。本节将介绍几种实用的统计分析工具。2.6.1节是Excel统计分析工具介绍,2.6.2节介绍SPSS统计分析工具,2.6.3节会对SAS统计分析工具做简单的了解,2.6.4节简要介绍统计分析R语言。

2.6.1 Excel统计分析工具

Microsoft Excel是微软公司的办公软件Microsoft Office重要的组件之一,它可以进行各种数据的处理、统计分析和辅助决策操作,当前已经广泛地应用于管理、统计财经、金融等众多领域。本节简要介绍Excel的统计分析工具。

Excel在默认情况下没有将统计分析模块展示在快速访问工具栏,所以首先需要找到它的统计分析工具。

在菜单栏右击,找到自定义快速访问工具栏,然后选择“加载项”,如图2-8所示。

单击“转到”按钮,出现如图2-9所示界面,然后勾选“分析工具库”,单击“确定”按钮后即可在“数据”菜单栏看到“数据分析”这一模块,如图2-10所示。

由图2-10可以看到,可以用Excel对数据进行方差分析、相关系数分析、傅里叶分析和画直方图等一系列操作,如果数据集较小,Excel统计分析工具是既方便又实用的选择。

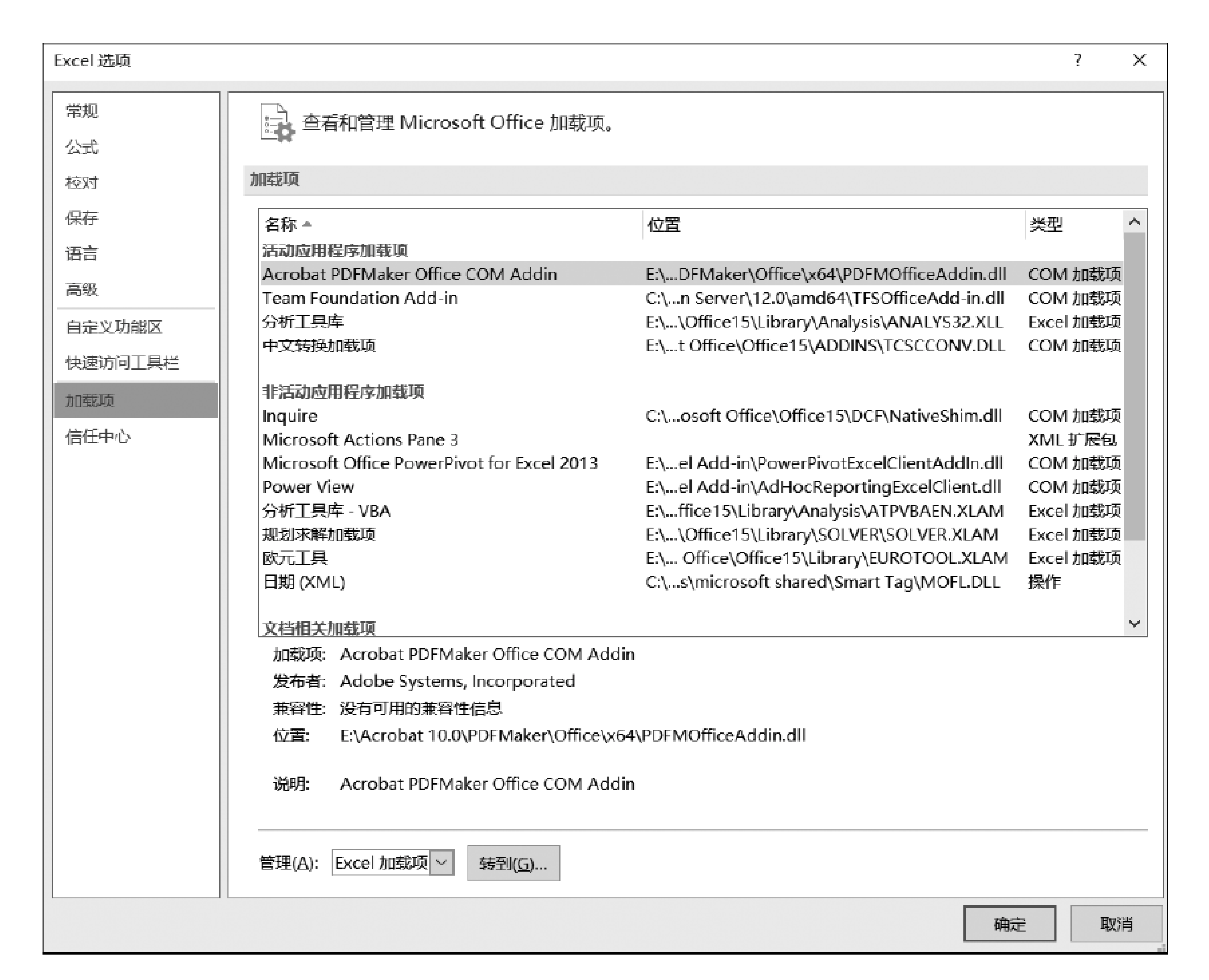


图 2-8　选择“加载项”

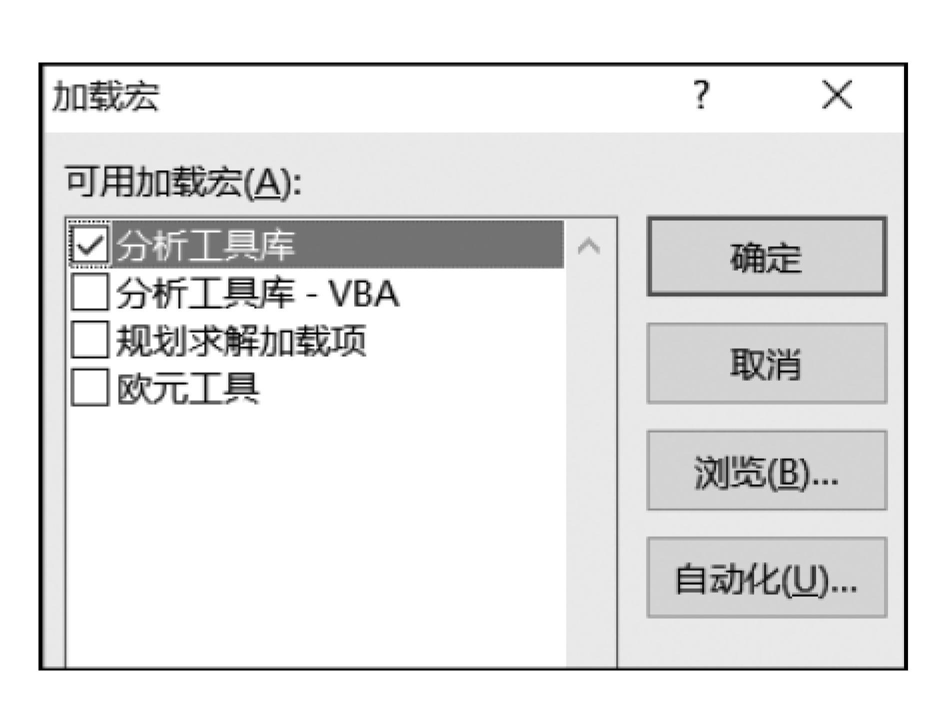


图 2-9　“加载宏”对话框

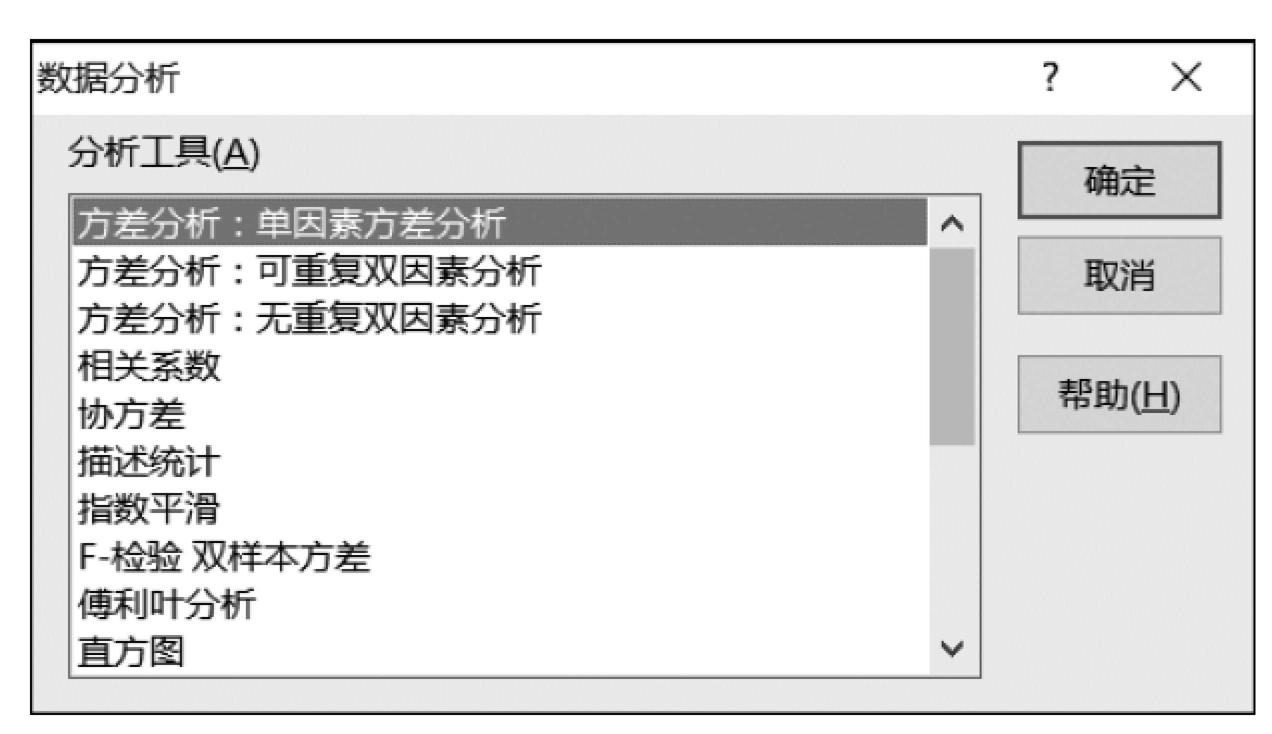


图 2-10　“数据分析”对话框

2.6.2 SPSS 统计分析工具

SPSS(Statistical Product and Service Solutions,统计产品和服务解决方案)是世界著名的商用统计分析软件之一,经过近 40 年的发展,在全球已拥有大量的用户。1984 年 SPSS 总部推出了世界上第一个统计分析软件微机版本 SPSS/PC+,开创了 SPSS 微机系列产品的开发方向,极大地扩充了它的应用范围,并使其能很快地应用于自然科学、技术科学、社会科学的各个领域。SPSS 有很多优点,它操作简便,功能强大,具有很强的针对性,而且能非常美观地展现统计分析的结果。

下面简要介绍 SPSS 的界面及主要功能,希望读者对 SPSS 能有一个直观的认识。

图 2-11　变量视图

启动 SPSS 后,可以看到它的主界面,如图 2-11 所示,其上方的菜单栏有 11 个选项,每一个选项都能完成特定的功能:

"文件"菜单用于新建 SPSS 各种类型的文件、打开文件或从其他数据源读入文件。

"编辑"菜单包含撤销、剪切、复制、粘贴、查找、更改 SPSS 设置等操作。

"视图"菜单主要用于显示或隐藏状态行、工具栏、网格线、值标签以及改变字体等。

"数据"菜单用于对 SPSS 数据文件进行操作,如定义变量、合并文件、转置变量和记录或产生分析的观测值子集等。

"转换"菜单在数据文件中对所选择的变量进行变换,并在已有的变量值的基础上计算新的变量。

"分析"菜单用于进行各种统计分析,包括各种统计过程,如回归分析、相关分析、因子分析等。

"直销"菜单内涵一些市场营销相关功能。

"图形"菜单可以用于产生条形图、饼图、直方图、散点图或其他高分辨率的图形以及动态的交互式图形。

“实用程序”菜单可以显示数据文件和变量的信息，定义子集，运行脚本程序，自定义SPSS菜单等。

SPSS的主界面就是它的编辑窗口。对于SPSS而言，它把数据编辑窗口分成了两张表显示，一张是数据视图，另一张则是变量视图。可以看到，图2-11中左下角有两个选项卡，单击按钮就可以切换两种视图，图2-11呈现的是变量视图。

数据视图可以直接由用户输入数据和存放数据，视图的左边显示的是个案的序号，上边显示的是变量的名称，变量视图主要是用来存放变量的，其中包括变量名称、变量类型、宽度、小数、标签、值、缺失、列、对齐和度量标准，前6个属性的定义如下。

名称：数据视图中变量的名字，必须以字母、汉字或者@开头，总长度不超过8个字符。

类型：变量的类型，一共有8种类型，如数值型、字符串型、时间型、逗号型等。

宽度：即变量取值所占有的宽度，默认为8位。

小数：即小数的位数，默认为2位。

标签：对变量名称的详细说明。

值：对变量取值的说明，可以类似Excel数据的下拉菜单选择，并且可以由数字代替具体的值，如图2-12所示，其中输入了性别属性的取值说明。



图2-12 “值标签”对话框

SPSS虽然操作简单，数据结果美观，但是在处理大量数据时速度并不是很理想。2.6.3节介绍一个对大量数据而言更快速的分析工具——SAS，它学习起来更为复杂，但是可以应用于大数据集的统计分析。

2.6.3 SAS统计分析工具

SAS (Statistical Analysis System)是一个模块化、集成化的大型应用软件系统。SAS系统主要完成以数据为中心的四大任务：数据访问、数据管理、数据呈现、数据分析。

SAS对于处理大数据具有很大优势，在金融领域SAS使用非常广泛。相对于SPSS来说，SAS有更加强大的绘图工具，而且可以编程，很受高端用户的欢迎，但是SAS是最

难掌握的统计分析软件之一。

图 2-13 即为 SAS 打开的界面，SAS 界面的主要窗口有 3 个，分别是编辑器、日志窗口和输出窗口。

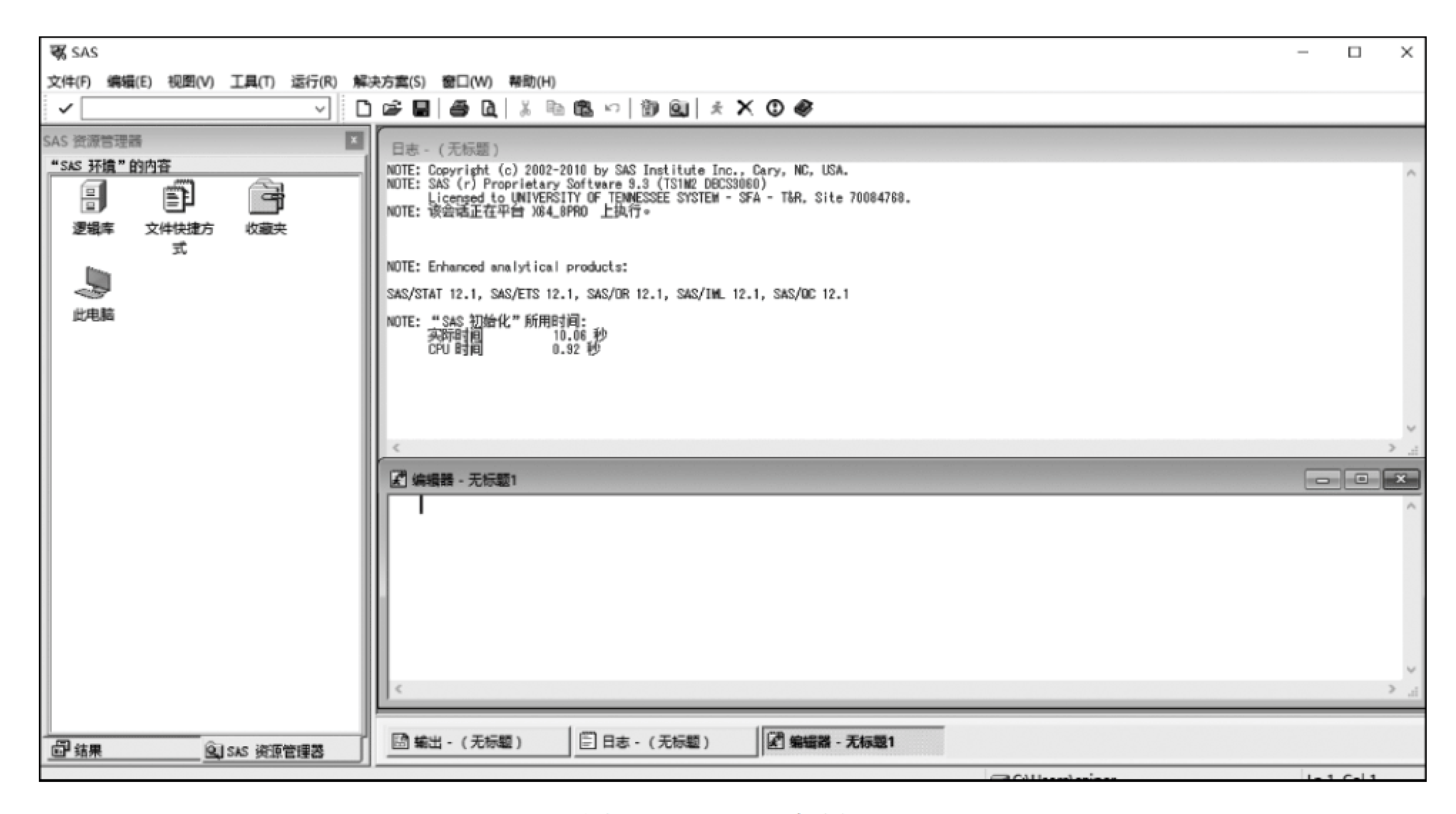


图 2-13　SAS 主界面

编辑器窗口是用来编写程序的，它在 SAS 中起着主导性作用，因此，有较好的编程基础才能更好地使用 SAS。

日志窗口是对编程后输出过程的分析，如果结果出错，日志里会以红色的文字标出，还有警告语句会以绿色的文字标出，蓝色是正常。查看日志有助于查找错误。

输出窗口是对输出结果的显示，所编程序的结果都是在该窗口输出。

可以看到，相比于 SPSS 来说，SAS 界面中的选项更少，这意味着需要自己编程实现大部分的工作。

SAS 程序由 SAS 数据步(DATA Step)和过程步(Proc Step)组成。数据步是用 DATA 语句开始的一组 SAS 语句，其作用是输入数据并建立 SAS 数据集。SAS 数据集的后缀名一律为.sd2，并不出现在程序中。SAS 系统只能分析 SAS 数据集的数据。SAS 数据步中的 input 和 cards 语句是数据步中的专用语句。其中 input 语句用来生成变量，cards 语句用来指明数据输入的开始。

过程步是用 PROC 语句开始的一组或几组 SAS 语句，其作用是激活 SAS 程序，并对已形成的 SAS 数据集通过过程步中的语句进行统计分析、打印等处理。SAS 程序中的字符串或数字之间均以空格隔开，并以“;”作为行结束符。

如图 2-14 为一个简单的 SAS 程序及其日志输出。

虽然 SAS 功能十分强大，但也因其高昂的价格而让一些中小企业望而却步。对于如今朝气蓬勃的互联网行业来说，它们更喜欢使用开源的软件。

```
日志 - (无标题)
      CPU 时间           0.01 秒


48    PROC  means;
49    VAR x y z;
50    RUN;

NOTE: 正在写入 HTML Body(主体)文件: sashtml.htm
NOTE: 数据集 WORK.CASE 中没有观测。
NOTE: "PROCEDURE MEANS" 所用时间(总处理时间):
      实际时间           1.32 秒
      CPU 时间           0.18 秒
```

```
编辑器 - 无标题1 *
DATA case;
INPUT x y z
CARDS
10  15  11
13  18  19
16  24  20
;
PROC  means;
VAR x y z;
RUN;
```

图 2-14　日志输出

2.6.4　R 语言统计分析工具

R 诞生于 1980 年左右，是 S 语言的一个分支，在统计领域广泛使用，可以认为 R 语言是 S 语言的一种实现，而 S 语言是由 AT&T 贝尔实验室开发的一种用来进行数据检索、统计分析和作图的解释型语言。

R 语言拥有一套完整的数据处理、计算和制图软件，其功能包括：数据存储和处理系统；数组运算工具(其向量、矩阵运算方面功能尤其强大)；完整连贯的统计分析工具；优秀的统计制图功能；简便而强大的编程语言既可以操纵数据的输入和输出，也可以实现分支和循环结构。而最重要的是 R 语言是完全免费开源的，所以对于很多中小型公司来说，R 语言是数据分析的首选工具。

安装好 R 语言之后，就可以打开其自带的 RGui 开始使用。其界面如图 2-15 所示。

在实际应用中，一般不会直接使用 Rgui，而是使用更美观、交互更人性化的 RStudio，其启动后界面如图 2-16 所示。

RStudio 提供的辅助功能有助于初学者顺利地输入函数，比如忘记画图函数 plot，输入前几位字母，如 pl，再按 Tab 键，会出现所有已安装的程序包中以 pl 开头的函数及简要介绍，按回车键即可选择。同时 Tab 键还可以显示函数的各项参数。例如，输入 plot(，RStudio 会自动补上右括号)，接着按 Tab 键则显示 plot()的各项参数。上下键可以用来切换上次运行的函数，RStudio 中 Ctrl+↑键则可以显示出最近运行的函数历史列表，如果想重复运行前面刚进行的命令，使用该操作非常方便。

可以用两种方式来编写 R 程序：命令行和脚本文件。



图 2-15　RGui 界面

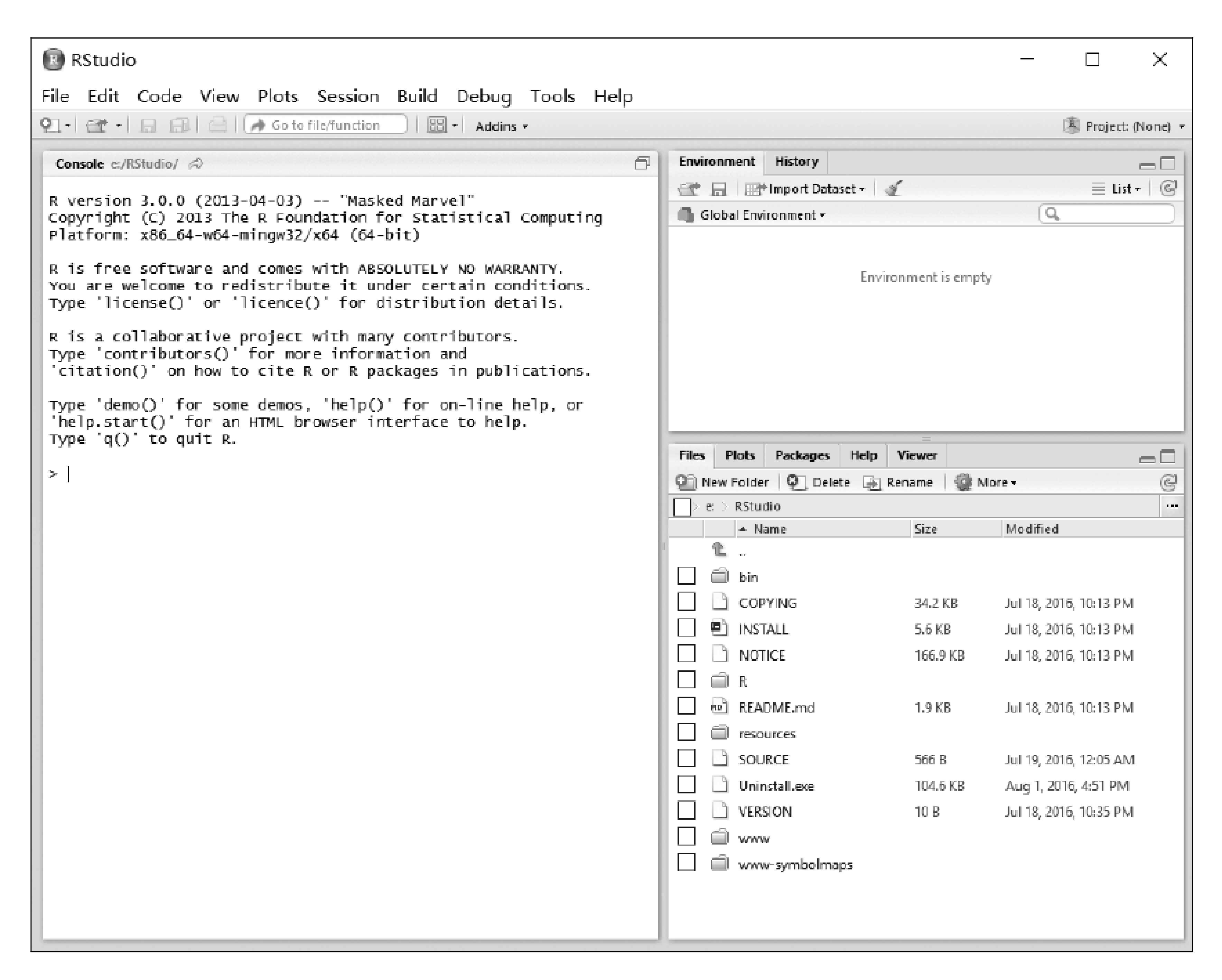


图 2-16　RStudio 界面

图 2-17 是 R 程序的命令行界面，在这个界面下，输入一行命令后按回车键就可以得到相应的输出。图 2-17 中进行了一个简单的加法运算。也可以通过脚本的形式编辑代码。在菜单栏选择 File→New→R script 可以建立一个 R 脚本文件。可以在这个文件里预先编辑一系列的代码，然后再执行这个脚本文件。更详细的 R 语言的使用将在第 8 章

中介绍。

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 2+1
[1] 3
> |
```

图 2-17 命令行

2.7 SPSS 案例分析

本节对美国某银行雇员状况数据集进行简单的 SPSS 数据处理与分析，包括数据清洗与转换、方差分析、相关性分析以及数据间距离的分析。在 2.7.1 节，首先进行待分析数据的准备，2.7.2 节介绍数据的录入与编辑方法，2.7.3 节是数据的清洗与转换相关内容的实现，在 2.7.4 节，将对数据进行简单的方差分析，而在 2.7.5 节，开展数据的相关性分析实现，最后数据间距离的分析将在 2.7.6 节探讨。

2.7.1 日志文件数据准备

一般来说，获取数据的途径是数据库、数据仓库或者已有的文件。在本节的例子中，从网上下载了美国某银行雇员状况数据集，该数据集包含了 474 个雇员的身份、工资等信息。具体属性描述见表 2-8，图 2-18 为 xls 格式的数据集。

表 2-8 银行雇员数据集属性描述

id	gender	bdate	educ	jobcat	salary	salbegin	jobtime	prevexp	minority	age
雇员编号	性别	生日	受教育程度	职务分类	当前工资	起始工资	受雇月数	过去经验	是否少数民族	年龄

2.7.2 数据录入与编辑

有了数据集，接下来需要将数据录入 SPSS 中。在建立数据文件之前，首先要定义变量，这需要打开如图 2-11 所示的变量视图窗口，进行变量的定义。按顺序依次定义变量名、变量类型、变量标签、变量取值标签、缺失值、变量显示格式以及变量的测度类型。图 2-19 是对本数据集包含的所有属性定义的变量。

其中 id 定义为数值类型，gender 定义为字符串。当所有变量都设定完毕后，就建立起了数据文件的框架，接下来就是将数据录入数据文件。

id	gender	bdate	educ	jobcat	salary	salbegin	jobtime	prevexp	minority	age	filter_$
1	m	03-Feb-1952	15	3	$57,000	$27,000	98	144	0	52	0
2	m	23-May-1958	16	1	$40,200	$18,750	98	36	0	46	1
3	f	26-Jul-1929	12	1	$21,450	$12,000	98	381	0	74	1
4	f	15-Apr-1947	8	1	$21,900	$13,200	98	190	0	57	1
5	m	09-Feb-1955	15	1	$45,000	$21,000	98	138	0	49	1
6	m	22-Aug-1958	15	1	$32,100	$13,500	98	67	0	45	1
7	m	26-Apr-1956	15	1	$36,000	$18,750	98	114	0	48	1
8	f	06-May-1966	12	1	$21,900	$9,750	98	0	0	38	1
9	f	23-Jan-1946	15	1	$27,900	$12,750	98	115	0	58	1
10	f	13-Feb-1946	12	1	$24,000	$13,500	98	244	0	58	1
11	f	07-Feb-1950	16	1	$30,300	$16,500	98	143	0	54	1
12	m	11-Jan-1966	8	1	$28,350	$12,000	98	26	1	38	1
13	m	17-Jul-1960	15	1	$27,750	$14,250	98	34	1	43	1
14	f	26-Feb-1949	15	1	$35,100	$16,800	98	137	1	55	1
15	m	29-Aug-1962	12	1	$27,300	$13,500	97	66	0	41	1
16	m	17-Nov-1964	12	1	$40,800	$15,000	97	24	0	39	1
17	m	18-Jul-1962	15	1	$46,000	$14,250	97	48	0	41	1
18	m	20-Mar-1956	16	3	$103,750	$27,510	97	70	0	48	0
19	m	19-Aug-1962	12	1	$42,300	$14,250	97	103	0	41	1
20	f	23-Jan-1940	12	1	$26,250	$11,550	97	48	0	64	1
21	f	19-Feb-1963	16	1	$38,850	$15,000	97	17	0	41	1
22	m	24-Sep-1940	12	1	$21,750	$12,750	97	315	1	63	1
23	f	15-Mar-1965	15	1	$24,000	$11,100	97	75	1	39	1
24	f	27-Mar-1933	12	1	$16,950	$9,000	97	124	1	71	1
25	f	01-Jul-1942	15	1	$21,150	$9,000	97	171	1	61	1
26	m	08-Nov-1966	15	1	$31,050	$12,600	96	14	0	37	1
27	m	19-Mar-1954	19	3	$60,375	$27,480	96	96	0	50	0
28	m	11-Apr-1963	15	1	$32,550	$14,250	96	43	0	41	1
29	m	28-Jan-1944	19	3	$135,000	$79,980	96	199	0	60	0
30	m	17-Sep-1961	15	1	$31,200	$14,250	96	54	0	42	1
31	m	24-Feb-1964	12	1	$36,150	$14,250	96	83	0	40	1
32	m	28-Jan-1954	19	3	$110,625	$45,000	96	120	0	50	0
33	m	18-Mar-1961	15	1	$42,000	$15,000	96	68	0	43	1
34	m	02-Feb-1949	19	3	$92,000	$39,990	96	175	0	55	0
35	m	22-Aug-1961	17	3	$81,250	$30,000	96	18	0	42	0
36	f	07-Aug-1963	8	1	$21,250	$11,250	96	52	0	40	1

图 2-18　数据集

名称	类型	宽度	小数	标签	值	缺失	列	对齐	测量
id	数值	4	0	雇员编号	无	无	3	右	名义(N)
gender	字符串	1	0	性别	{f, 女}...	无	6	左	名义(N)
bdate	日期	10	0	生日	无	无	13	右	度量
educ	数值	2	0	受教育程度(年)	{0, 0 (Missi...	0	5	右	有序(O)
jobcat	数值	1	0	职务分类	{0, 0 (缺失)}...	0	6	右	有序(O)
salary	美元	8	0	当前工资	{$0, missing...	$0	8	右	度量
salbegin	美元	8	0	起始工资	{$0, missing...	$0	8	右	度量
jobtime	数值	2	0	受雇月数	{0, missing}...	0	8	右	度量
prevexp	数值	6	0	过去经验(月)	{0, missing}...	无	8	右	度量
minority	数值	1	0	是否少数民族	{0, 否}...	9	8	右	有序(O)
age	数值	8	0	年龄	无	无	8	右	度量

图 2-19　变量定义对话框

单击数据视图按钮，可以看到以上所定义的变量都在表的顶端显示。这时，只需要将每个实体的数据对应的变量值一一输入就可以了。图 2-20 显示了输入的第一个实体数据。

如果数据集比较大，手动输入变量的工作量将会非常大，如果数据集是 xls 格式的，就可以使用 SPSS 的数据导入功能一次性导入所有数据。当然，除了 xls 格式数据，SPSS

id	gender	bdate	educ	jobcat	salary	salbegin	jobtime	prevexp	minority	age
1	m	02/03/1952	15	3	$57,000	$27,000	98	144	0	52

图 2-20　输入个案数据

还可以导入 txt、csv、dat 等格式的数据。

SPSS 采用类似 Excel 表格的形式组织数据，可以轻松地对数据进行增、删、改、查等操作。

2.7.3　数据清洗与转换

1. 数据清洗

观测值的缺失往往会给统计分析带来许多麻烦，尤其是在时间序列分析中更是如此。时间序列里如果存在缺失值，可能导致一些变量的计算不能进行，比如在计算环比和定基发展速度的时候，假如时间序列里有缺失值，由于系统自动将数据文件中数值型变量的缺失值视为 0，因此计算中有可能会出现除数为 0 的情况。除了时间序列中的缺失值，其他变量类型的缺失值也会对统计分析结果造成一定的影响。如图 2-21 所示，可以发现 id 为 43 的实体的变量 salary 是一个缺失值，像这样的缺失值在数据集中还有很多。产生缺失值的原因可能是录入数据时的失误，或者采集数据时的失误，或者其他原因。

id	gender	bdate	educ	jobcat	salary	salbegin	jobtime	prevexp	minority	age
43	m	01/18/1964	12	1	.	$14,250	95	46	0	40

图 2-21　缺失值

选择菜单栏“分析”→“缺失值分析”命令，然后选择想要分析缺失值的变量名，单击确定后，就可以得到如图 2-22 所示的缺失值分析报告。从表中，可以看到变量 salary 有 6 个缺失值，占总数的 1.3%，而 salbegin 有 3 个缺失值，占总数的 0.6%。并且系统自动求出了这两个变量的平均值和标准偏差。

单变量统计

	N	平均值	标准 偏差	缺失		极值数目[a]	
				计数	百分比	低	高
salary	468	$34,349.04	$17,037.075	6	1.3	0	56
salbegin	471	$16,994.79	$7,847.526	3	.6	0	59

a. 超出范围 (Q1 - 1.5*IQR, Q3 + 1.5*IQR) 的个案数。

图 2-22　缺失值分析

SPSS 提供了替换缺失值功能。在菜单栏选择“转换”→“替换缺失值”命令，可以看到如图 2-23 所示的界面。

选择需要替换缺失值的变量，然后选择替换方法。SPSS 提供了 5 种替换策略，分别是序列平均值、邻近点平均值、邻近点中位数、线性插值以及邻近点线性趋势。这里为 salary 和 salbegin 两个变量选择的替换方法是取邻近点的平均值。如图 2-24 所示，id 为

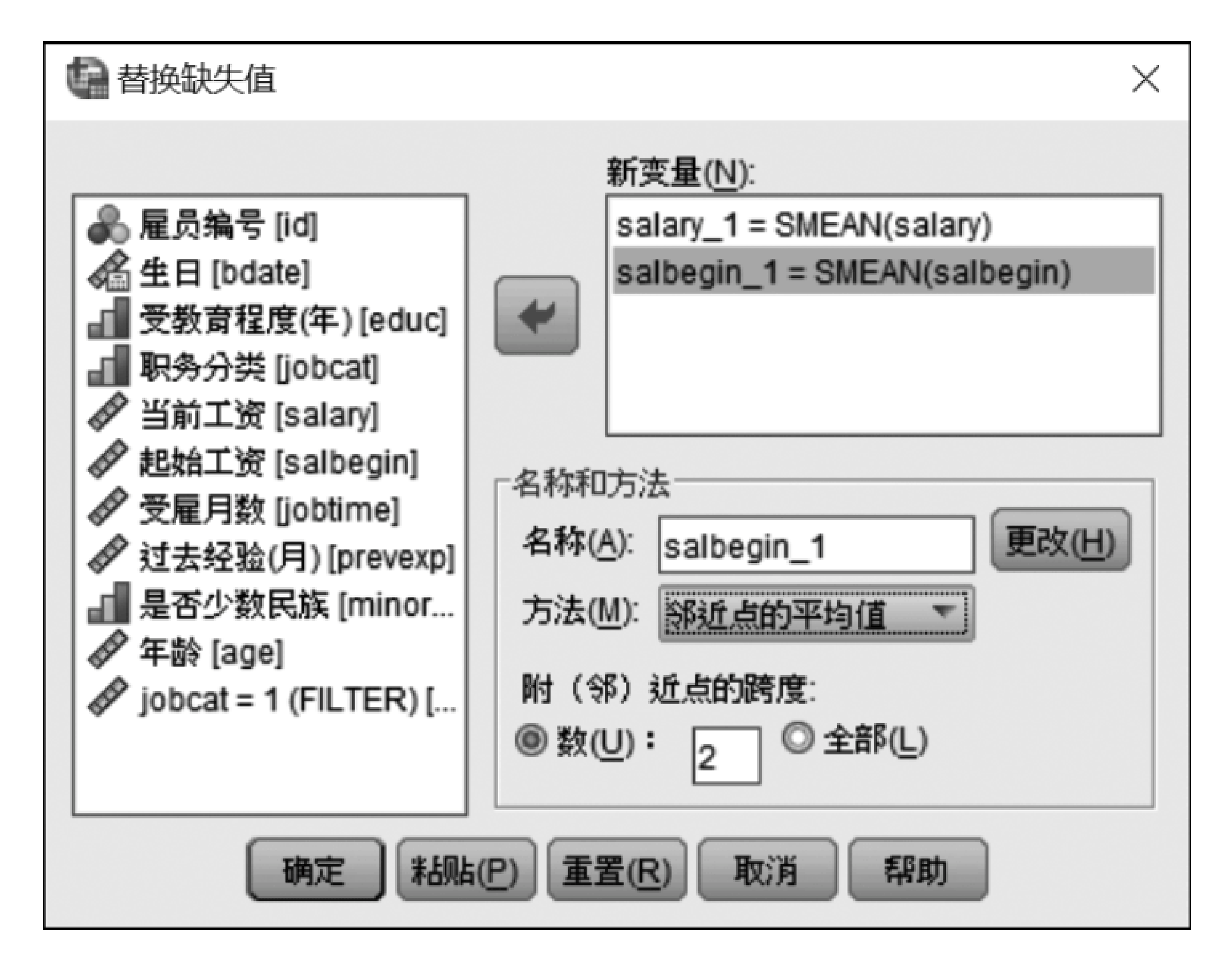


图 2-23 “替换缺失值”对话框

43 的 salary 变量值被替换为＄34 349.0。

	educ	jobcat	jobtime	prevexp	minority	age	filter_$	salary	salbegin
43	12	1	95	46	0	40	1	$34,349.0	$14,250.0

图 2-24 替换结果

再次运行缺失值分析，可以看到如图 2-25 所示的结果，salary 和 salbegin 变量已经没有缺失值了，并且变量的平均值和标准偏差也发生了变化。

单变量统计

	N	平均值	标准 偏差	缺失		极值数目[a]	
				计数	百分比	低	高
salary	474	$34,349.038	$16,928.6721	0	.0	0	63
salbegin	474	$16,994.788	$7,822.6003	0	.0	0	61

a. 超出范围 (Q1 - 1.5*IQR, Q3 + 1.5*IQR) 的个案数。

图 2-25 缺失值替换后的分析结果

2. 数据转换

在做统计分析时，有时候需要对变量进行转换以达到更好的效果。例如，在回归分析中经常需要对变量做对数化处理。SPSS 提供了强大的变量计算功能，新变量的计算可以借助计算变量功能来完成。

如果希望将受教育程度（educ）转换成受教育等级（educ_lv），选择菜单栏中的“转换”→“计算变量”，出现如图 2-26 所示的界面，选择 educ_lv 变量，在“数字表达式”输入框中输入计算公式“((5-MOD(educ,5))＋educ)/5”，变量计算结果如图 2-27 所示。之后可

以单击“类型与标签”按钮，设定这个转换后的新变量的类型和标签。



图 2-26 “计算变量”对话框

id	gender	bdate	educ	educ_lv	jobcat	jobtime	salary	salbegin	prevexp	minority
1	m	02/03/1952	15	4	3	98	$57,000.0	$27,000.0	144	0
2	m	05/23/1958	16	4	1	98	$40,200.0	$18,750.0	36	0

图 2-27 变量计算结果对话框

通过上述公式，受教育程度按 5 年划分，转换为相应的受教育等级，小于 5 年的为 1 级，小于 10 年且大于等于 5 年的为 2 级，以此类推。其实，可以看出，这相当于是对受教育程度做了一个离散化操作。

2.7.4 数据方差分析

单因素方差分析用来测试某一个控制变量的不同水平是否给观察变量造成显著差异和变动。本节将考察不同的受教育程度是否对起始工资的差异造成影响。

在菜单栏选择“分析”→“比较平均值”→“单因素 ANOVA”命令，打开“单因素方差分析”对话框。如图 2-28 所示，分别选择因变量与因子，并单击“对比”按钮。如图 2-29 所示，输入系数 1、0、0、−1 表示对比第一组和最后一组。



图 2-28 “单因素方差分析”对话框

单击“继续”按钮，选择选项。如图 2-30 所示，分别选择描述性、方差同质性检验以及平均值图。回到初始界面后，单击“确定”按钮，得到分析结果，如图 2-31 所示。



图 2-29 “单因素 ANOVA：对比”对话框

单因素 ANOVA: 选项

Statistics

- ☑ 描述性(D)
- ☐ 固定和随机效果(F)
- ☑ 方差同质性检验(H)
- ☐ Brown-Forsythe
- ☐ Welch

☑ 平均值图(M)

缺失值

- ◉ 按分析顺序排除个案(A)
- ○ 按列表排除个案(L)

继续 取消 帮助

图 2-30 “单因素 ANOVA：选项”对话框

图 统计描述

从图 2-31 可以看出，SPSS 对每组数据进行了统计描述，包括每组的平均值、标准偏差、标准错误以及平均值 95%置信区间的下限与上限值。可以看出明显的结论：受教育程度越高，就越有可能拿到更高的起始工资。

图 2-32 给出了方差分析结果，其中包括组间、组内的偏差平方和、均方、F 值和概率 P 值，从显著性水平的 $P<0.05$ 看出，各组间均值在 0.05 水平上有显著性差异。

ANOVA

起始工资

	平方和	df	均方	F	显著性
组之间	7928095205	3	2642698402	59.100	.000
组内	2.102E+10	470	44715381.33		
总计	2.894E+10	473			

图 2-32 方差分析表

图 2-33 是以受教育等级为横轴，以起始工资的平均值为纵轴的散点图，从图中可以直观地看出各组均值的分布。



2.7.5 数据相关性分析

相关分析(correlation analysis)是研究对象之间是否存在某种依存关系及相关程度的一种统计方法。在统计学上,两个连续型变量的关系多以线性关系进行分析,线性关系分析是用直线方程的原理来估计两个变量关系的强度,比如常见的相关系数就是刻画两个变量线性相关关系的指标,相关系数越大表示线性关系越强,相关系数越小表示线性关系越弱。本节对起始工资与当前工资之间的相关性进行分析。

选择"分析"→"相关"→"双变量"打开双变量相关性窗口,如图 2-34 所示,选择起始工资与当前工资添加入变量框。选择 Pearson 相关系数,点击确定得到分析结果如图 2-35 所示。

从图 2-35 的结果看出,当前工资与起始工资相关系数达到 0.868,说明这两个变量之间具有较强的相关性,并且呈正相关。



图 2-34 "双变量相关性"对话框

2.7.6 数据间距离分析

在统计学领域或者数据挖掘领域,通常用距离来描述样本之间的相似关系。一般将样本视为 P 维空间的点,并在该空间定义点与点之间的距离,将距离较近的点归为一类,距离较远的点视为属于不同的类。

本节将选择 educ(受教育程度)、jobtime(受雇月数)、salary(当前工资)、salbegin(起始工资)、prevexp(过去经验)这 5 个变量作为距离测量的 5 个维度。

相关性

		当前工资	起始工资
当前工资	Pearson 相关性	1	.868**
	显著性（双尾）		.000
	N	474	474
起始工资	Pearson 相关性	.868**	1
	显著性（双尾）	.000	
	N	474	474

**. 在置信度（双测）为 0.01 时，相关性是显著的。

图 2-35　相关性分析结果

首先在菜单栏选择“分析”→“相关”→“距离”，打开如图 2-36 所示的“距离”对话框，可以看到，上述 5 个变量已经添加进了变量栏。在计算距离栏内选择“个案间”，即计算观测量之间的距离。测量栏选择不相似性，系统默认使用 Euclidean 距离测度观测量或变量的不相似性。单击“测量”按钮，进入如图 2-37 所示的“距离：非相似性测量”窗口，度量标准选择“区间”单选按钮下的“测量”列表框中的“Euclidean 距离”。因为 salary 与 salbegin 数值普遍很大，为了不过于影响距离测度，应对所有的变量进行标准化处理。如图 2-37 所示，在“标准化”栏里选择“范围 0 至 1”，则系统自动将每个观测量值减去它们的最小值，然后除以极差，将它们标准化到 0～1 之间。



图 2-36　“距离”对话框



图 2-37 “距离：非相似性测量”对话框

设置完成后单击“确定”按钮，进入计算分析。运行完毕后，会输出一个 474×474 的矩阵，如图 2-38 所示，该矩阵是一个对称矩阵，包含了所有数据点对之间的距离。

	1	2	3	4	5	6	7	8	9	10	11	12
1	.000	.301	.659	.651	.132	.325	.220	.539	.322	.458	.279	.674
2	.301	.000	.809	.716	.233	.142	.184	.374	.226	.556	.242	.631
3	.659	.809	.000	.506	.607	.705	.626	.801	.607	.289	.595	.809
4	.651	.716	.506	.000	.593	.603	.579	.506	.563	.328	.629	.349
5	.132	.233	.607	.593	.000	.212	.096	.447	.191	.381	.159	.617
6	.325	.142	.705	.603	.212	.000	.128	.288	.107	.443	.183	.547
7	.220	.184	.626	.579	.096	.128	.000	.375	.108	.379	.114	.581
8	.539	.374	.801	.506	.447	.288	.375	.000	.341	.516	.446	.319
9	.322	.226	.607	.563	.191	.107	.108	.341	.000	.358	.112	.570
10	.458	.556	.289	.328	.381	.443	.379	.516	.358	.000	.380	.553
11	.279	.242	.595	.629	.159	.183	.114	.446	.112	.380	.000	.666
12	.674	.631	.809	.349	.617	.547	.581	.319	.570	.553	.666	.000
13	.382	.144	.767	.632	.279	.079	.192	.255	.171	.499	.245	.540
14	.234	.231	.578	.563	.102	.156	.056	.398	.095	.339	.088	.593
15	.423	.342	.665	.407	.332	.236	.274	.158	.255	.376	.352	.321
16	.427	.318	.759	.478	.356	.251	.305	.140	.307	.472	.399	.316
17	.287	.117	.766	.649	.214	.127	.176	.330	.210	.508	.255	.562
18	.430	.552	1.023	.977	.528	.638	.594	.807	.681	.846	.654	.914
19	.330	.346	.611	.398	.263	.259	.248	.284	.264	.335	.337	.369
20	.457	.347	.701	.432	.364	.243	.300	.114	.273	.414	.376	.313

图 2-38 距离矩阵

以上就是本节 SPSS 分析实例的所有内容，对于实际应用来说，掌握这些是远远不够的，本节内容希望起到一个抛砖引玉的效果。

2.8 小　结

- **数据分析**的标准流程分为以下几步：数据分析问题的理解，相关数据的理解，相关数据的准备，分析模型的建立，分析模型的评估，分析结果的部署。
- 一般来说，**准确性**、**完整性**、**一致性**、**时效性**等指标是评价数据质量最常用的标准。
- 获取的**数据集**都由一个一个**数据对象**组成，每一个对象都代表一个实例。**属性**是一个字段，表示数据对象的一个特征。
- **标称属性**的值是符号或事物的名字，其中每个值代表某种类别、编码或状态。
- **二元属性**是仅有两种可能的状态(0 或 1，真或假)的标称属性。如果两个状态同等重要，则它是对称的，否则是非对称的。
- **序值属性**是其可能值之间具有有意义的序或排位，但相继值之间的量值未知的属性。
- **数值属性**是最常用的一种数据类型，它是可度量的，用整数或实数值表示，它定量地描述对象。
- **数据的中心趋势度量**是指一组数据向某一中心值靠拢的倾向，测度中心趋势就是要寻找数据一般水平的代表值或中心值。中心趋势度量一般包括数据的中值、中位数、均值与中列数。
- **数据的离散趋势度量**反映了数据集中的值远离其中心值的程度，因此也可以叫做离中趋势度量，主要有极差、分位数、五数概括、方差和标准差等几种度量方法。
- 对象**相似性**和**相异性**度量用于聚类分析、离群点分析、最近邻分类等数据挖掘应用中。度量方法包括 Jaccard 系数、欧几里得距离、曼哈顿距离、切比雪夫距离、闵可夫斯基距离、汉明距离、编辑距离以及余弦相似性。
- 关于**数据相关性**，本章主要介绍了皮尔逊相关系数、斯皮尔曼秩相关系数及协方差。
- **数据清洗**试图填补缺失值，光滑噪声，同时识别离群点，并纠正数据的不一致性。
- **数据集成**将来自多个源的数据整合成一致的数据存储。该过程中需要解决异常数据、冗余数据及重复数据。
- **数据归约**包括维归约、数量归约和数据压缩。**维归约**减少样本空间中所包含的属性个数。其方法包括小波变换、主成分分析以及属性子集选择，前面两种是把原数据变换或投影到维数较小的样本空间中，而后者是通过相关性等方法分析后，检测样本空间中不相关、弱相关或冗余的属性或维，然后予以删除。**数量归约**用替代的、较小的数据集表示形式替换原数据集，包括参数方法或非参数方法。参数方法就是为数据集拟合一个描述模型来估计数据，使得存储形式只需要存放模型参数，而不是实际的数据集，例如各种回归模型；非参数方法包括直方图、聚类和抽样等。**数据压缩**使用不同变换方法以得到原数据的压缩形式。无损压缩用于要求重构的信号与原始信号完全一致的场合。一些常用的无损压缩算法有赫夫曼算法和 LZW 压缩算法。如果只能近似重构或不完全恢复原数据，则该数据

归约技术称为有损压缩。有损压缩广泛应用于语音、图像和视频数据的压缩。

2.9 习　　题

1. 获取数据的主要途径有哪些？

2. 数据质量的评价标准有哪些？请分别简要描述。

3. 有一组已经排序的数据如下：5,10,11,13,15,35,50,55,72,92,204,215,如果采用等频分箱将这些数据划分为 4 个箱,15 在第几个箱子内？

4. 请分别介绍均值、中位数和截断均值在反映数据中心方面的特点。

5. 计算数据 16,17,19,22,23,24,29,33,34,39 的均值、中位数、四分位数和标准差。

6. 已知某班 20 名学生的数学成绩分别为 64,65,67,70,71,75,75,75,76,77,78,79,79,79,80,81,82,84,85,90。

(1) 求该组成绩的中位数、众数和极差。

(2) 画出该组成绩的盒图。

7. 在数据挖掘中为什么要对原始数据进行预处理？数据预处理中填补空缺值的方法有哪些？

8. 计算数值属性相似性的常用方法有哪些？它们各有何优缺点？

9. 设 $\boldsymbol{x}$、$\boldsymbol{y}$ 为向量。

(1) $\boldsymbol{x}=(1,1,1,1)$,$\boldsymbol{y}=(2,2,2,2)$,求余弦相似度和欧氏距离。

(2) $\boldsymbol{x}=(0,1,0,1)$,$\boldsymbol{y}=(1,0,1,0)$,求余弦相似度和欧氏距离。

(3) $\boldsymbol{x}=(2,-1,0,2,0,-3)$,$\boldsymbol{y}=(-1,1,-1,0,0,-1)$,求余弦相似度。

10. 给定两个向量对象,分别表示为 $p_1(1,3,1,5,1)$ 和 $p_2(2,4,6,8,9)$。

(1) 计算两个对象之间的欧氏距离。

(2) 计算两个对象之间的曼哈顿距离。

(3) 计算两个对象之间的切比雪夫距离。

11. 给定 3 个向量对象,分别表示为 $p_1(0,0)$、$p_2(1,0)$ 和 $p_3(0,2)$。

(1) 计算两两间的欧氏距离。

(2) 计算两两间的曼哈顿距离。

(3) 计算两两间的切比雪夫距离。

(4) 计算两两间的闵可夫斯基距离(以变参数为 2 的欧氏距离计算)。

12. 计算(1,1,0)、(1,-1,0)和(-1,1,0)两两之间的 Jaccard 相似度。

13. 简要说明数据准备的主要步骤有哪些。

14. 在数据清洗中,处理数据缺失的方法有哪些？如何去掉数据的噪声？

15. 简述主成分分析的具体步骤。

16. 常用的抽样方法有哪些？

17. 有数据值 2000,3000,4000,6000,10000。对这些数据进行规范化。

(1) 令 min=0,max=1,进行最小-最大规范化。

(2) 进行 z 分数规范化。

18．R 语言有哪些功能？

2.10 参考文献

[1] 张文霖．一切从数据准备开始[J]．数据，2013(8)：48-49．
[2] 简祯富，许嘉裕．大数据分析与数据挖掘[M]．北京：清华大学出版社，2016．
[3] JIAWEI HAN(加)．数据挖掘概念与技术[M]．北京：机械工业出版社，2006．
[4] 杨青云，赵培英，杨冬青，等．数据质量评估方法研究[J]．计算机工程与应用，2004，40(09)：3-4．
[5] 刘明吉，王秀峰，黄亚楼．数据挖掘中的数据预处理[J]．计算机科学，2000，27(04)：54-57．
[6] 菅志刚，金旭．数据挖掘中数据预处理的研究与实现[J]．计算机应用研究，2004，21(07)：117-118．
[7] 杨辅祥，刘云超，段智华．数据清理综述[J]．计算机应用研究，2002，19(03)：3-5．
[8] 吴爱华．不一致数据的查询处理[D]．上海：复旦大学，2010．
[9] 康睿智，郝文宁．数据归约效果评估方法研究[J]．计算机工程与应用，2016,53(15)：93-96．
[10] 蔡维玲，陈东霞．数据规范化方法对K近邻分类器的影响[J]．计算机工程，2010，36(22)：175-177．
[11] Friston K J，Frith C D，Liddle P F，et al．Functional Connectivity：the Principal-Component Analysis of Large (PET) Data Sets．[J]．Journal of Cerebral Blood Flow & Metabolism Official Journal of the International Society of Cerebral Blood Flow & Metabolism，1993，13(1)：5-14．
[12] 田兵．单因素方差分析的数学模型及其应用[J]．阴山学刊：自然科学版，2013，27(2)：24-27．



数据可视化

当我们准备好数据，知道了数据由什么类型的属性或字段组成以后，就可以通过相应的数据预处理技术填补数据集中的缺失值，光滑噪声，识别离群点，由基本的统计描述方法获得关于数据集属性值的中心趋势和离散趋势。但是，是否可以用相关的方法对整理好的数据集进行可视化的观察，直观地了解数据看上去如何？值如何分布？离群点分布的大概位置？数据对象之间大致的相似性如何？借助于数据可视化技术不仅可以图形化的方式观察数据，更有助于识别隐藏在杂乱数据集中的关系、趋势和偏差。不同的应用场景和数据类型使用不同的可视化方法。例如，简单的表格类型数据可以使用柱状图和折线图等，而复杂的层级结构数据可以使用树图等可视化方法。

本章从可视化技术的应用及划分开始(3.1 节)，在本章的其余部分，将学习到最常用的高维数据的可视化方法(3.2 节)和网络数据的可视化方法(3.3 节)，最后通过两个竞赛案例对可视化技术的实际应用做详细介绍(3.4 节)。

3.1 可视化简介

数据可视化是一门古老的学科。早在公元 2 世纪，人们就开始用行和列来管理数据。到中世纪时期，人们已开始使用包含等值线的地磁图、表示海上主要风向的天象图等。在 20 世纪，随着计算技术和显示技术的发展，数据可视化技术更是得到了飞速发展。近十多年来，随着大数据技术的发展，现有可视化技术已难以应对海量、高维、多源和动态数据可视化的挑战，数据可视化技术随之迎来了新的发展机遇。

数据可视化是被许多学科和领域专家所使用的视觉传递现代技术，它包含了数据可视表达的创造性和研究性两个层面。数据可视化将数据所包含的信息的综合体，包含属性和变量，抽象化为一些图表形式。数据可视化的主要目的是借助统计图、散点图和其他图的形式准确、清晰、有效地传达数据中所包含的信息。有效的可视化更能进一步帮助用户分析数据，推论事件，寻找规律。它使得复杂数据更容易被用户所理解和使用。

数据可视化处理的对象是数据，根据处理对象的不同，数据可视化可分为科学可视化和信息可视化两个分支。又由于数据可视化具有分析推理等数据分析的功能，且数据分析对理解数据和使用数据具有非常重要的作用，将可视化与数据分析相结合，又形成了一个新的数据可视化方向，即可视分析。

科学可视化是一个跨学科的科学分支,也是可视化领域发展最早和最成熟的一个学科。科学可视化主要关注三维空间数据的可视化,强调线、面、体等几何、拓扑结构的真实表达,其主要应用领域是自然科学,如物理、化学、医学、流体力学、生物学、气象学等学科。根据数据的不同类别,科学可视化可分为标量场可视化、矢量场可视化和张量场可视化。

信息可视化是以增强人的认知能力为目的的抽象数据和非结构化数据可视表达研究。与科学可视化相比,信息可视化主要关注抽象数据,不仅包括数值数据,也包括非数值数据,如文本、图像、层次结构等。统计信息可视化起源于统计图形学,与信息图形学、视觉设计等现代技术相关。信息可视化的研究融合了多个学科,如计算机科学、计算机图形学、人机交互、可视设计、心理学、商业方法等。在本章,主要关注与大数据关系最密切的信息可视化内容。

可视分析是一门辅以交互可视界面进行分析推理的科学,主要通过耦合人的分析能力与机器的计算能力智能地解决由于数据量、问题的复杂性等带来的棘手难题,可视交互界面则是耦合人与机器的介质。可视分析融合了数据表达与分析、人机交互和可视化等技术,已经并仍在推动分析决策、技术转移等技术的发展。

数据可视化是艺术也是科学。作为一门科学,可视化应该真实反映数据所包含的信息;作为一门艺术,可视化应该具有艺术美感。为了实现可视化在科学与艺术间的平衡,可视化需要达到真、善、美三种境界:

真,即真实性,指可视化结果应正确反映数据的本质。

善,即易感知,指可视化结果应有利于公众认知数据背后所蕴含的现象和规律。

美,即艺术性,指可视化结果的形式和内容应和谐统一。

在对数据进行可视化之前首先要了解数据类型,不同的数据类型都有对应的可视化方法。在3.2节中介绍针对高维数据的可视化方法,在3.3节中介绍针对网络数据的可视化方法。

3.2 高维数据可视化

在本节中主要介绍高维数据可视化的处理分析过程。在对高维数据可视化时,如果维度过高则需要对数据进行降维处理,在3.2.1中介绍几种常见的降维处理方法,在3.2.2节中介绍几种常用的高维数据可视化方法。

无论是在日常生活中还是在科学研究中,高维数据处处可见。例如,一件简单的商品就包含了型号、厂家、价格、性能、售后服务等多种属性;再如,在癌症研究中,为了找到与致癌相关的基因,需要综合分析不同病人的成百上千个基因表达;对大气、海洋、宇宙等复杂物理现象的计算模拟,也要考虑到诸如温度、压强等多个维度因素。人们一般很难直观快速地理解三维以上的数据,而将数据转化为可视的形式,就可以帮助人们理解和分析高维空间中的数据特性。高维数据可视化技术旨在用图形表现高维度的数据,并辅以交互手段,帮助人们分析和理解高维数据。

高维数据可视化主要分为降维方法和非降维方法。降维方法指将高维数据投影到低

维空间，尽量保留高维空间中原有的特性和聚类关系。常见的降维方法有主成分分析(Principle Component Analysis，PCA)、多维尺度分析(Multi-Dimensional Scaling，MDS)、自组织图(Self-Organization Map，SOM)等。这些方法通过数学方法将高维数据降维，进而在低维屏幕空间中显示。通常，数据在高维空间中的距离越近，在投影图中两点的距离也越近。高维投影图可以很好地展示高维数据间的相似度以及聚类情况等，但并不能表示数据在每个维度上的信息，也不能表现维度间的关系。高维投影图损失了数据在原始维度上的细节信息，但直观地提供了数据之间宏观的结构。

图 3-1 中包括了大多数流形学习方法，不过这里面没有 t-SNE，相比于其他算法，t-SNE 算是比较新的一种方法，也是效果比较好的一种方法。

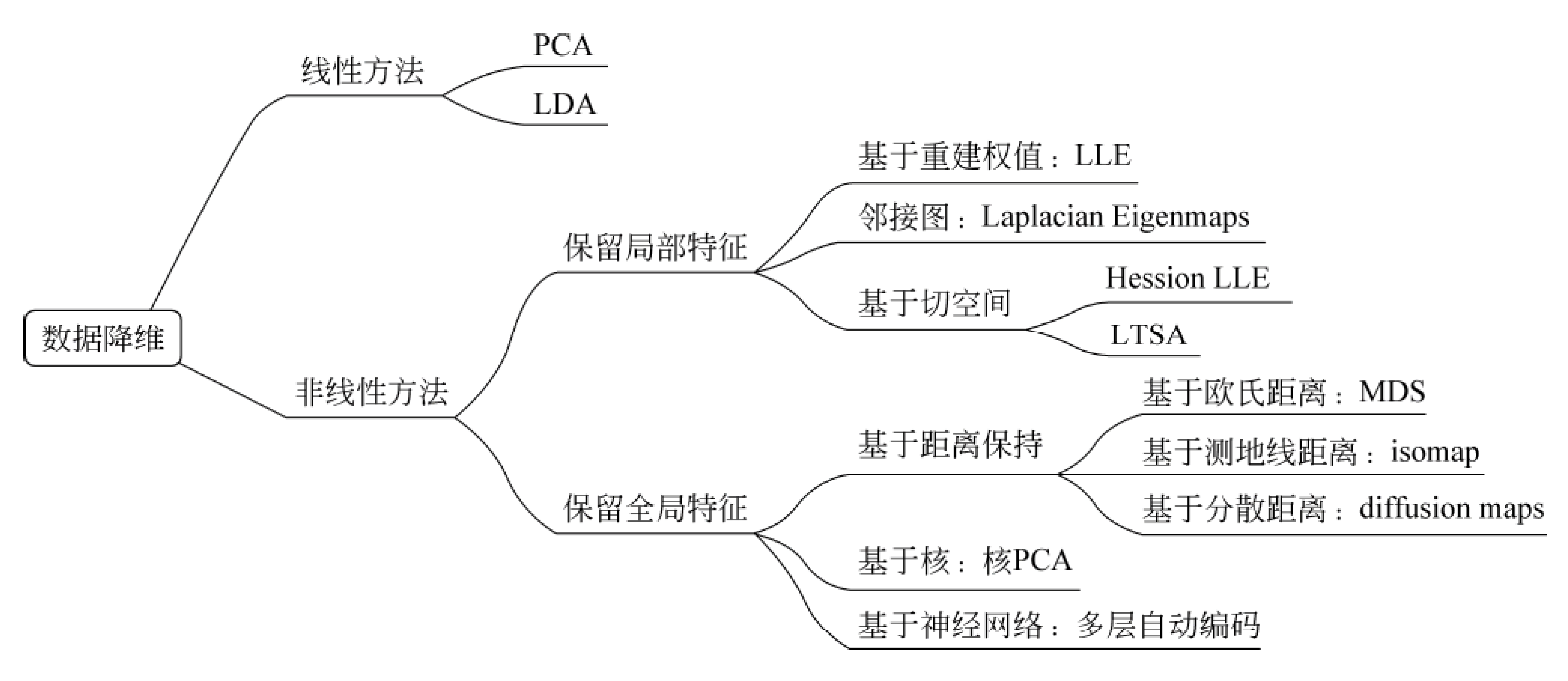


图 3-1　数据降维的分类

3.2.1　降维方法

1. PCA

主成分分析(PCA)是一种较为普遍的线性降维方法，它的目标是通过某种线性投影将高维的数据映射到低维的空间中表示，并期望在所投影的维度上数据的方差最大，以此使用较少的数据维度，同时保留较多的原数据点的特性。

PCA 的原理就是将原来的样本数据投影到一个新的空间中，相当于矩阵分析中将一组矩阵映射到另外的坐标系下。通过一个转换坐标，也可以理解成把一组坐标转换到另外一组坐标系下，但是在新的坐标系下，表示原来的样本不需要那么多的变量，只需要原来样本的最大的一个线性无关组的特征值对应的空间的坐标即可。通俗地理解，就是如果把所有的点都映射到一起，那么几乎所有的信息(如点和点之间的距离关系)都丢失了，而如果能使映射后方差尽可能地大，那么数据点则会分散开来，以此来保留更多的信息。可以证明，PCA 是丢失原始数据信息最少的一种线性降维方式(实际上就是最接近原始数据，但是 PCA 并不试图去探索数据内在结构)。

设 n 维向量 $\boldsymbol{w}$ 为目标子空间的一个坐标轴方向(称为映射向量)最大化数据映射后的方差，有

$$\max_{w} \frac{1}{m-1} \sum_{i=1}^{m} (\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}))^2$$

其中，m 是数据实例的个数，$\boldsymbol{x}_i$ 是数据实例 i 的向量表达，$\bar{\boldsymbol{x}}$ 是所有数据实例的平均向量。定义 $\boldsymbol{W}$ 为包含所有映射向量为列向量的矩阵，经过线性代数变换，可以得到如下优化目标函数：

$$\min_{w} \operatorname{tr}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{W}), \text{ s.t. } \boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \boldsymbol{I}$$

其中，tr 表示矩阵的迹，$\boldsymbol{A}$ 是数据协方差矩阵：

$$\boldsymbol{A} = \frac{1}{m-1} \sum_{i=1}^{m} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^{\mathrm{T}}$$

容易得到最优的 $\boldsymbol{W}$ 是由数据协方差矩阵前 k 个最大的特征值对应的特征向量作为列向量构成的。这些特征向量形成一组正交基并且最好地保留了数据中的信息。

PCA 追求的是在降维之后能够最大化保持数据的内在信息，并通过衡量在投影方向上的数据方差的大小来衡量该方向的重要性。但是这样投影以后对数据的区分作用并不大，反而可能使得数据点揉杂在一起无法区分。这也是 PCA 存在的最大一个问题，这导致使用 PCA 在很多情况下的分类效果并不好。如图 3-2 所示，若使用 PCA 将数据点投影至一维空间上时，PCA 会选择 2 轴（ϕ2），这使得原本很容易区分的两簇点被揉杂在一起变得无法区分；而这时若选择 1 轴（ϕ1）将会得到很好的区分结果。

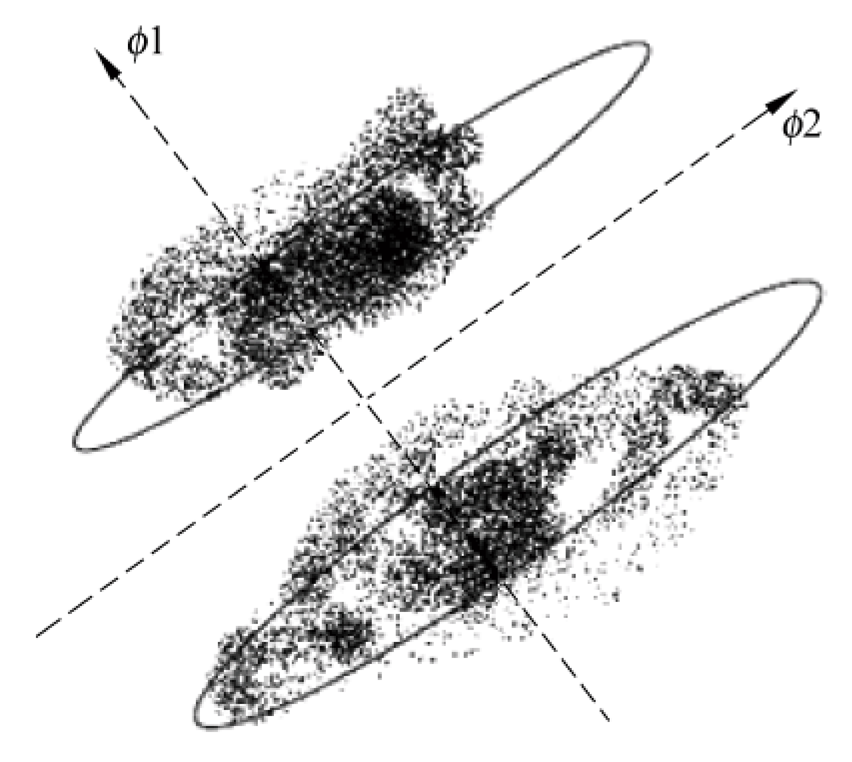


图 3-2　PCA 分析图

比如，原来的样本的维数是 30×1 000 000，就是说有 30 个样本，每个样本有 1 000 000 个特征点，这个特征点太多了，需要对这些样本的特征点进行降维。在降维的时候会计算一个原来样本矩阵的协方差矩阵，这里就是 1 000 000×1 000 000，当然，这个矩阵太大了，计算的时候会用其他的方式进行处理，这里只是讲解基本的原理。然后通过这个 1 000 000×1 000 000 的协方差矩阵计算它的特征值和特征向量，最后获得具有最大特征值的 特征向量构成转换矩阵。例如前 29 个特征值已经能够占到所有特征值的 99%以上，那么只需要提取前 29 个特征值对应的特征向量即可。这样就构成了一个 1 000 000×29 的转换矩阵，然后用原来的样本乘以这个转换矩阵，就可以得到原来的样本数据在新的特征空间的对应的坐标，将样本的维数降为 30×29，这样原来的训练样本中每个样本的特征值的个数就降到了 29 个。

一般来说，PCA 降维后的每个样本的特征的维数不会超过训练样本的个数，因为超出的特征是没有意义的。虽然目前比较火的图像、视频、文字等多媒体数据大多是在一个非线性的流形上或者附近，PCA 作为一种线性的降维方法可能很少使用了，但是 PCA 依然是不可替代的，比如 2015 年 IEEE VIS 年会中的可视分析（VAST2015）最佳论文就颁给了一个使用 PCA 来做动态网络分析的工作。

2. MDS

多维尺度分析(MDS)是分析研究对象的相似性或差异性的一种多元统计分析方法。MDS 主要考虑的是成对样本间的相似性,主要思想是用成对样本的相似性来构建合适的低维空间,使得样本在低维空间的距离和高位空间中的距离尽可能保持一致。根据样本是否可计量,可分为 Metric MDS 和 Nonmetric MDS。目标函数是每对低维空间中的欧氏距离与高维空间中相似度差的平方和,最小化目标函数,用一些数值优化方法来得到最优解。

3.2.2 非降维方法

非降维方法保留了高维数据在每个维度上的信息,可以展示数据的所有维度。各种非降维方法的主要区别在于如何对不同的维度进行数据到图像属性的映射。当维度数量较少时,可以直接通过与位置、颜色、形状等多种视觉属性相结合的方式对高维数据进行编码,例如,在形状、大小、颜色上映射数据维度的小图标方法,或用不同角度映射数据维度呈放射形状的星形图(star plot)。但当维度数量增多,数据量变大,或对数据呈现精度的需求提高时,这些方法往往难以满足需要。在处理科学、社会研究和应用中的复杂高维数据时,需要伸缩性(scalability)更强的高维数据可视化方法,包括散点图矩阵(scatterplot matrix)和平行坐标(parallel coordinates)等。

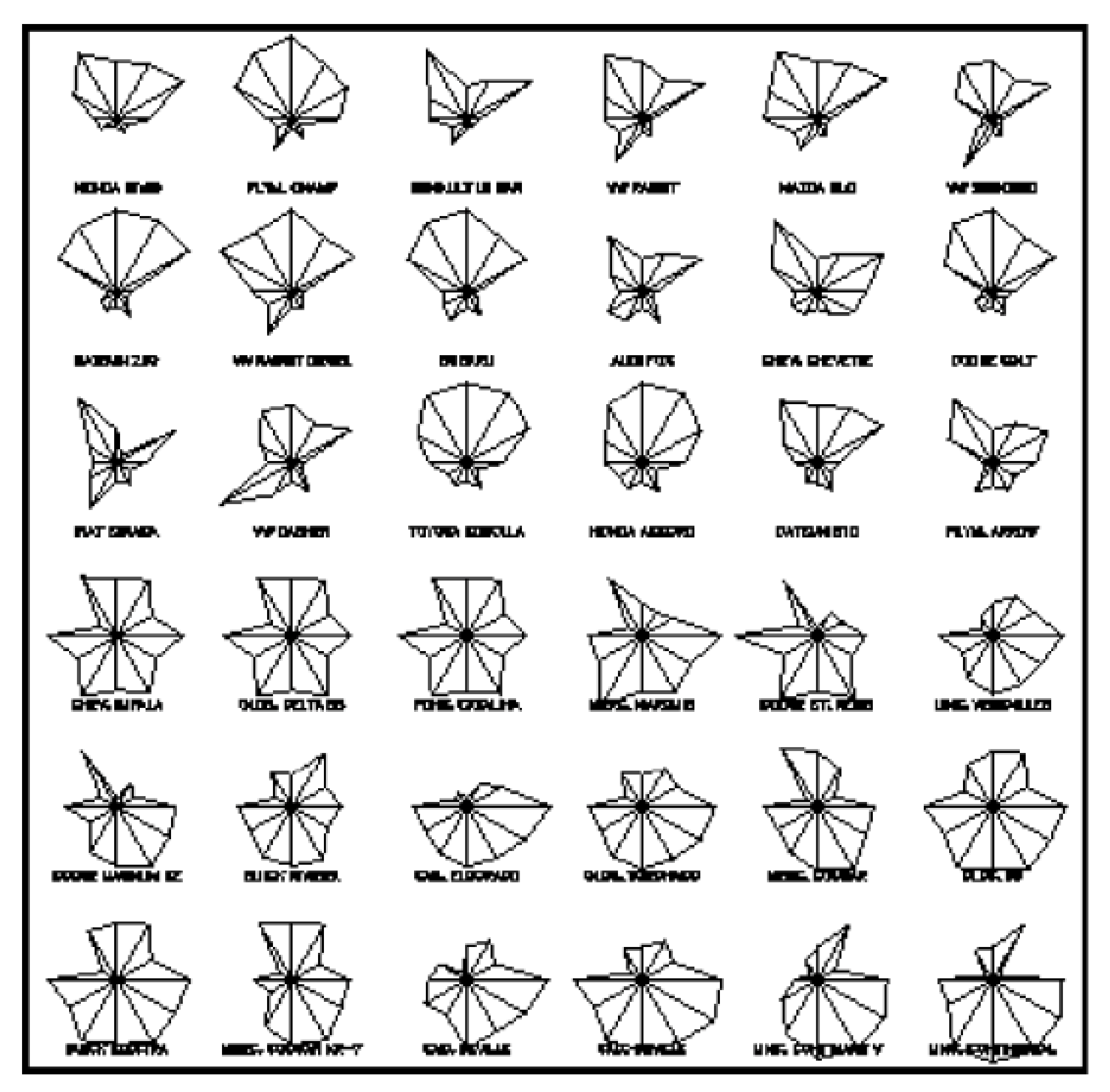
1. 星形图

星形图,又称为雷达图(radar chart)。星形图可以看成平行坐标的极坐标版本。多元数据的每个属性由一个坐标轴表示,所有坐标轴链接到共同的原点(圆心),其布局沿圆周等角度分布,每个坐标轴上的点的位置由数据对象的值与该属性最大值的比例决定,用折线连接所有坐标轴上的点,围成一个星形区域。星形区域的形状和大小反映了数据对象的属性。

星形图提供了一种比较紧凑的数据可视化。随着数据维度的增加,可视化所占的圆形区域内需要显示更多的坐标,但是,其总面积并不变。由于人类视觉识别对形状和大小的敏感性,星形图能使得不同数据对象之间的比较更加容易和高效。

图 3-3 左边展现了汽车数据 12 个数字变量的星形图;这 12 个变量按照图 3-3 右边的变量赋值键所示进行排列。图 3-3 右边的为星形图的变量赋值键;其侧面和底部的变量与大小相关联,其他变量与价格和性能相关联。图 3-3 中每颗星代表一种汽车模型;星中的每条射线长度与各个变量值成正比。对于汽车数据而言,一个变量有较大的值(即较长的射线)则可能意味着这是一辆好车,例如 PRICE、TURN 和 GRATIO 等变量。在图 3-3 中最显著的标示就是:在顶部的星形图中顶部的价格和性能(price and performance)变量射线较长,并且底部的尺寸(size)变量射线较短;但反过来这样理解也是对的:最重的模型在星形图的底部一行。

图 3-4 展现了美国 50 个州以及首都华盛顿特区的犯罪率。在右图的范例中,可以看



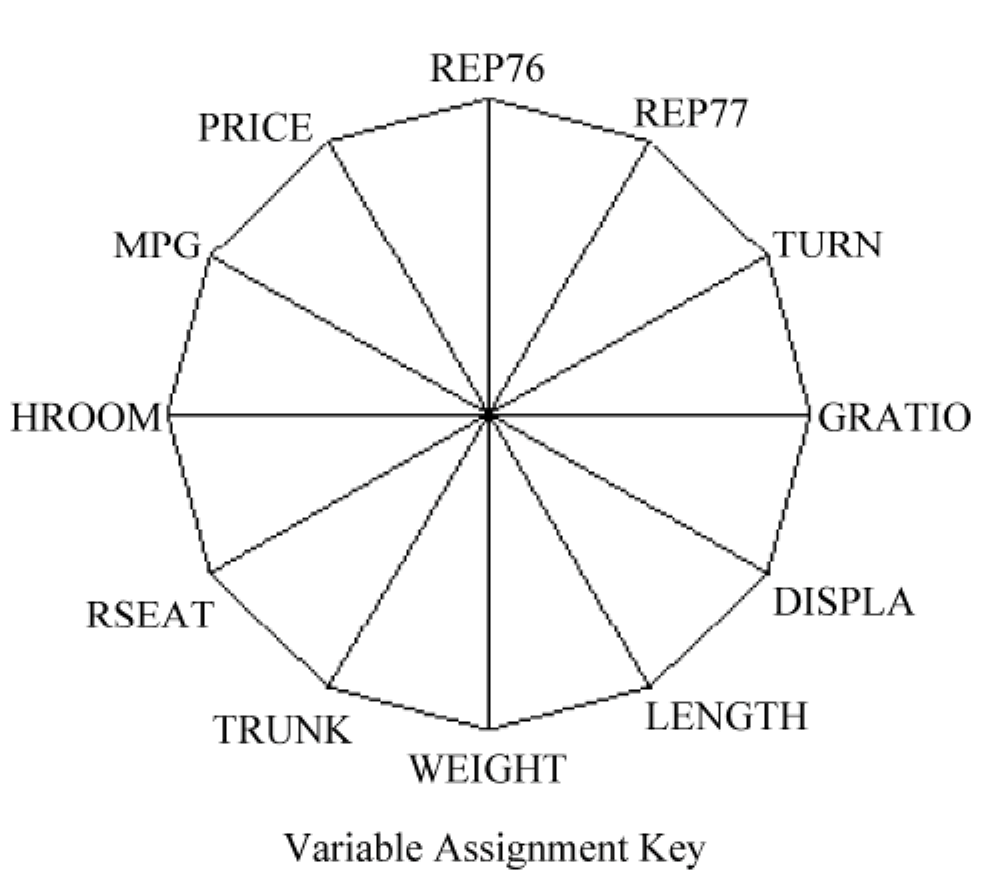


图 3-3 汽车数据星形图

图片来源：https：//andrewpwheeler. wordpress. com/2012/03/14/reference-lines-for-star-plots-aid-interpretation/

到在星形图中 7 个坐标分别表示 7 种类型的犯罪(车辆失窃、盗窃、抢劫、谋杀和强奸等)。范例中展示的乔治亚州除了强奸发生率低之外,其余各种犯罪率都较高。在左图中,代表所有 50 个州和华盛顿特区的星形图按照顺序依次排列,用户可以通过比较不同星形区域的大小和形状,了解各州的犯罪情况。例如,北达科他州是各类犯罪最少的州;而南达科他州除了强奸发生率较高之外,其余犯罪率都较低。

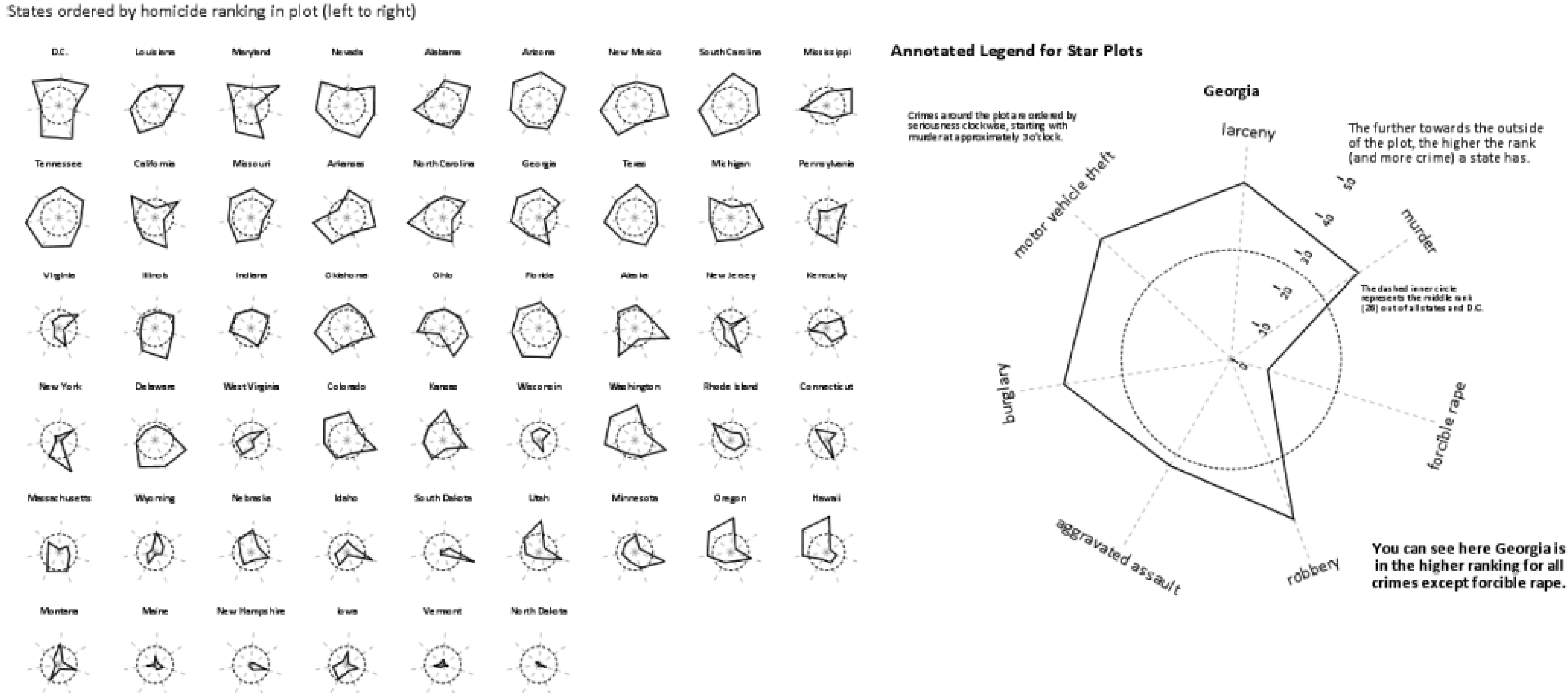


图 3-4 美国 50 个州以及首都华盛顿特区的犯罪率星形图

图片来源：https：//andrewpwheeler. wordpress. com/2012/03/14/reference-lines-for-star-plots-aid-interpretation/

2. 切尔诺夫脸谱图

切尔诺夫脸谱图(Chernoff Faces)是由美国统计学家切尔诺夫在 1976 年率先提出的,用脸谱来分析多维度数据,即将 P 个维度的数据用人脸部位的形状或大小来表示。此方法和星形图类似,也采用图标表示单个的多元数据对象,不同的是,切尔诺夫脸谱图采用模拟人脸的图标来表示数据对象,它可以把多元数据用二维的人脸的方式整体表现出来。各类数据变量经过编码后,转变为脸型、眉毛、眼睛、鼻子、嘴、下巴等面部特征,数据整体就是一张表情各异的人脸。例如,图 3-5 展示了使用切尔诺夫脸谱图可视化同样的美国各州犯罪率数据的例子,其中脸的长度表示谋杀案的发生率,脸的宽度表示强奸案的发生率,从图中可以明显地观察到阿拉斯加州(Alaska)的强奸案发生率较高,哥伦比亚特区(District of Columbia)的谋杀案发生率较高。

切尔诺夫脸谱图的灵感来源于,当人们面对错综复杂的信息时,人脑会自动过滤掉无用信息,保留有用信息。人脑通常可以察觉到一些非常细微甚至难于测量的变化,然后对其做出反应,同时,人脑区分脸谱时,这种优越性更加明显,因为无论是脸的胖瘦还是五官的大小和位置,都极易给人留下深刻的印象,因而易于区别。但人们对于人脸上各个部分或者特征的感知程度不同,所以需要根据数据分析的目的和属性的优先级别来选择合适的属性与某个人脸特征之间形成映射。

3. 散点图

散点图(scatterplot)的本质是将抽象的数据对象映射到二维的直角坐标系表示的空间。数据对象在坐标系的位置反映了其分布特征,直观、有效地揭示两个属性之间的关系。面向多元数据,散点图的思想可泛化为:采用不同的空间映射方法将多元数据对象布局在二维平面空间中,数据对象在空间中的位置反映了其属性及相互之间的关联,而整个数据集在空间中的分布则反映了各个维度之间的关系及数据集的整体特性。图 3-6 表达的是 2013 年世界各国预期寿命与人均国内生产总值的散点图可视化。每个数据对象(圆点)代表一个国家,圆点的大小表示国家人口,圆点的颜色表示国家所在的洲。

散点图矩阵是散点图的高维扩展,它从一定程度上克服了在平面上展示高维数据的困难,在展示多维数据的两两关系时有着不可替代的作用。散点图矩阵是由所有两两维度间的散点图按矩阵形式排列而成的,每个散点图都表现出两个维度间的关系。通过散点图矩阵,可以看到数据在任意两个维度间的相关特性以及聚类情况。它的缺点是不能显示各个数据在多个维度上的协同关系,同时需要很大的显示空间,需要的显示空间面积正比于维度数目的平方。

图 3-7 是以鸢尾花数据为例,利用 R 语言 graphics 包中的 pairs()函数绘制的散点图矩阵。图中不同颜色分别代表不同品种的鸢尾花,从图中可以观察到各类鸢尾花的花瓣、花萼长宽的大体分布以及它们两两之间的关系。

The Face of Crime in the United States

Violent Crime

HEIGHT OF FACE
Murder
HEIGHT OF MOUTH Aggravated Assault
SHAPE OF FACE Robbery
WIDTH OF FACE
Forcible Rape

Property Crime

WIDTH OF EYES
Motor Vehicle Theft
HEIGHT OF EYES
Larceny-Theft
WIDTH OF MOUTH
Burglary

United States
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming

SOURCE
US Statistical Abstract

BY
Nathan Yau
FlowingData
http://flowingdata.com

图 3-5 切尔诺夫脸谱图

图片来源：http：//flowingdata. com/2010/08/31/how-to-visualize-data-with-cartoonish-faces/crime-chernoff-faces-by-state-edited-2/

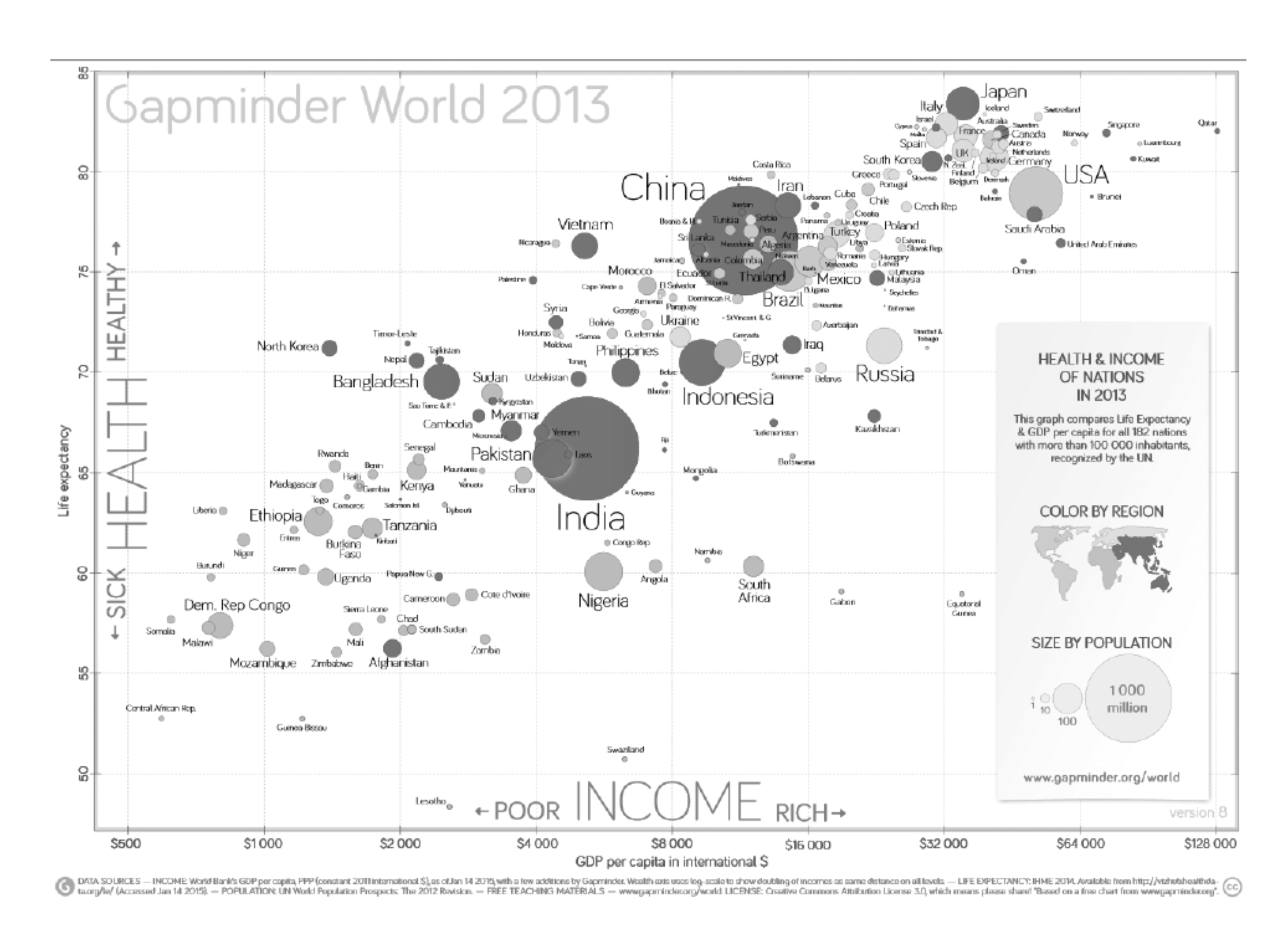


图 3-6　2013 年世界各国预期寿命与人均国内生产总值的散点图可视化

图片来源：http：//www. gapminder. org

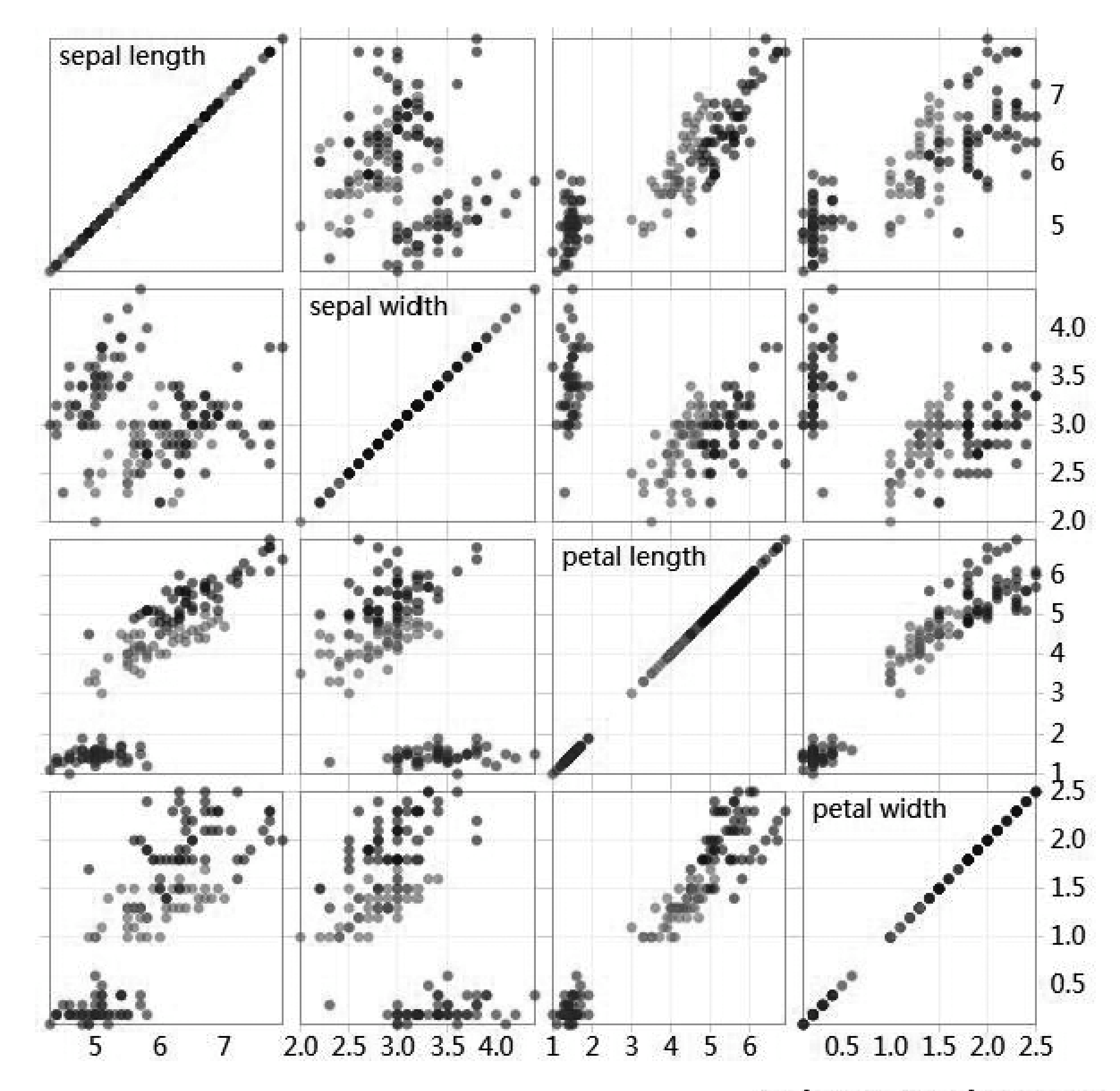


图 3-7　鸢尾花散点图矩阵

图片来源：https：//github. com/d3/d3/wiki/Gallery

4. 平行坐标

平行坐标(parallel coordinates)是将高维数据的各个变量维度用一系列相互平行的坐标轴来表示，变量值对应轴上的位置。将描述不同变量维度的同一数据对应各点连接成折线，代表一个数据的一条折线在平行的坐标轴上的投影就反映了变化趋势和各个变量维度间的相互关系。平行坐标能够帮助分析数据在多个维度上的分布和多个维度之间的关系，且平行坐标需要的显示面积仅正比于维度的数目。由于每个数据点都表现为一条折线，所以平行坐标在两个维度之间关系的表现上不如散点图清楚，并且当数据量增大时更容易受到图元堆叠的影响。

图 3-7 中的例子使用的鸢尾花数据也可以用平行坐标系表示，如图 3-8 所示。从图 3-8 中可以看到，鸢尾花的 Iris setosa 类花萼长度主要分布于 4.3～5.8cm 之间，花萼宽度主要分布于 2.9～4.4cm 之间，但也有一个例外，其花萼宽度为 2.3，这种情况可以猜测到是植物中的突变或者异常；花瓣长度主要分布于 1.0～1.9cm 之间，花瓣宽度主要分布于 0.1～0.6cm 之间；其他两种鸢尾花的情况也可以从图中观察到。除此之外，还可以从图中观察出它们两两之间更多的联系。

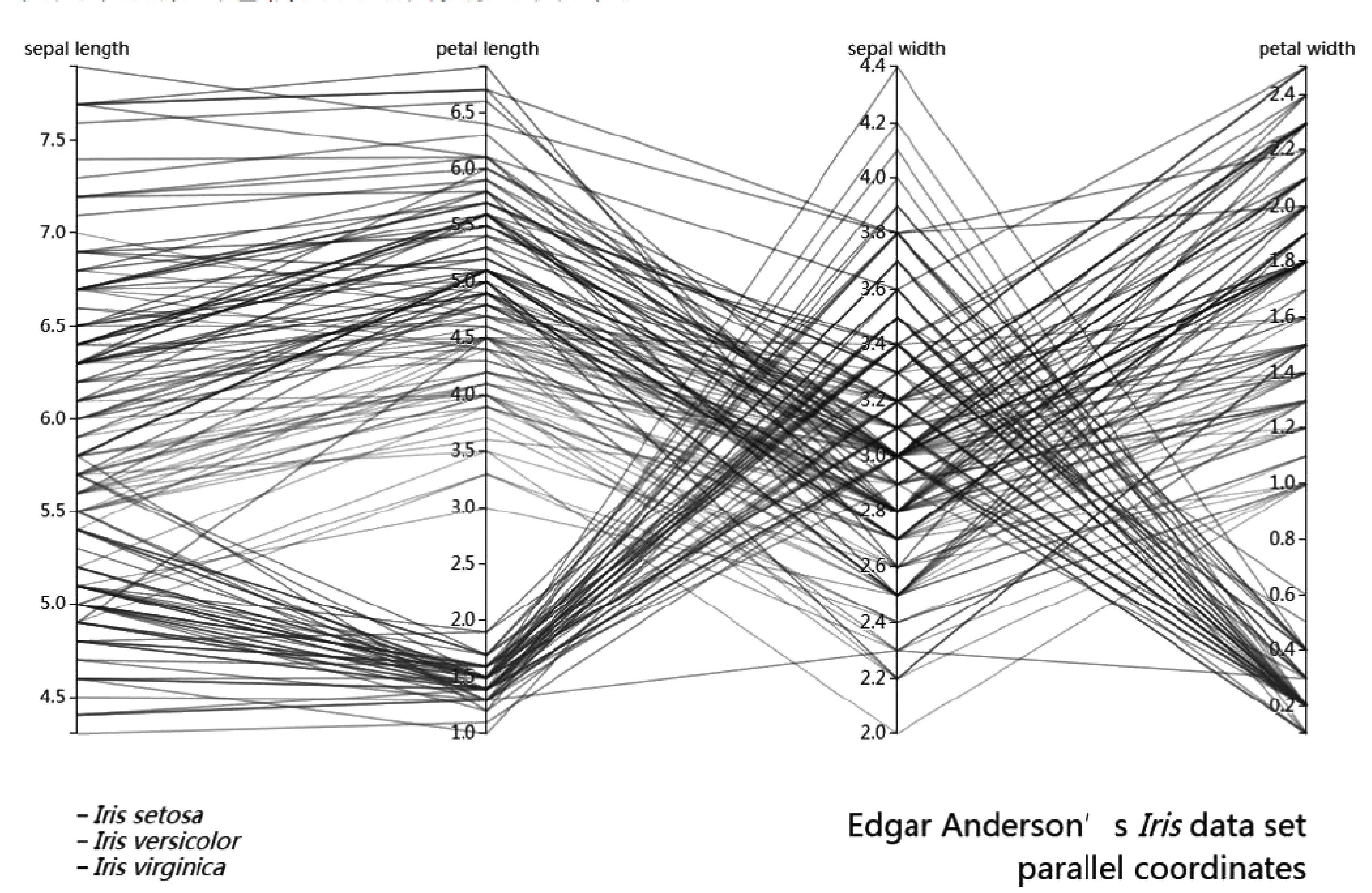


图 3-8 鸢尾花平行坐标系

图片来源：https://github.com/d3/d3/wiki/Gallery

图 3-9 为在平行坐标中结合散点图。图中的两个属性 Horsepower 和 Displacement 的关系在两个轴之间用散点图表示。

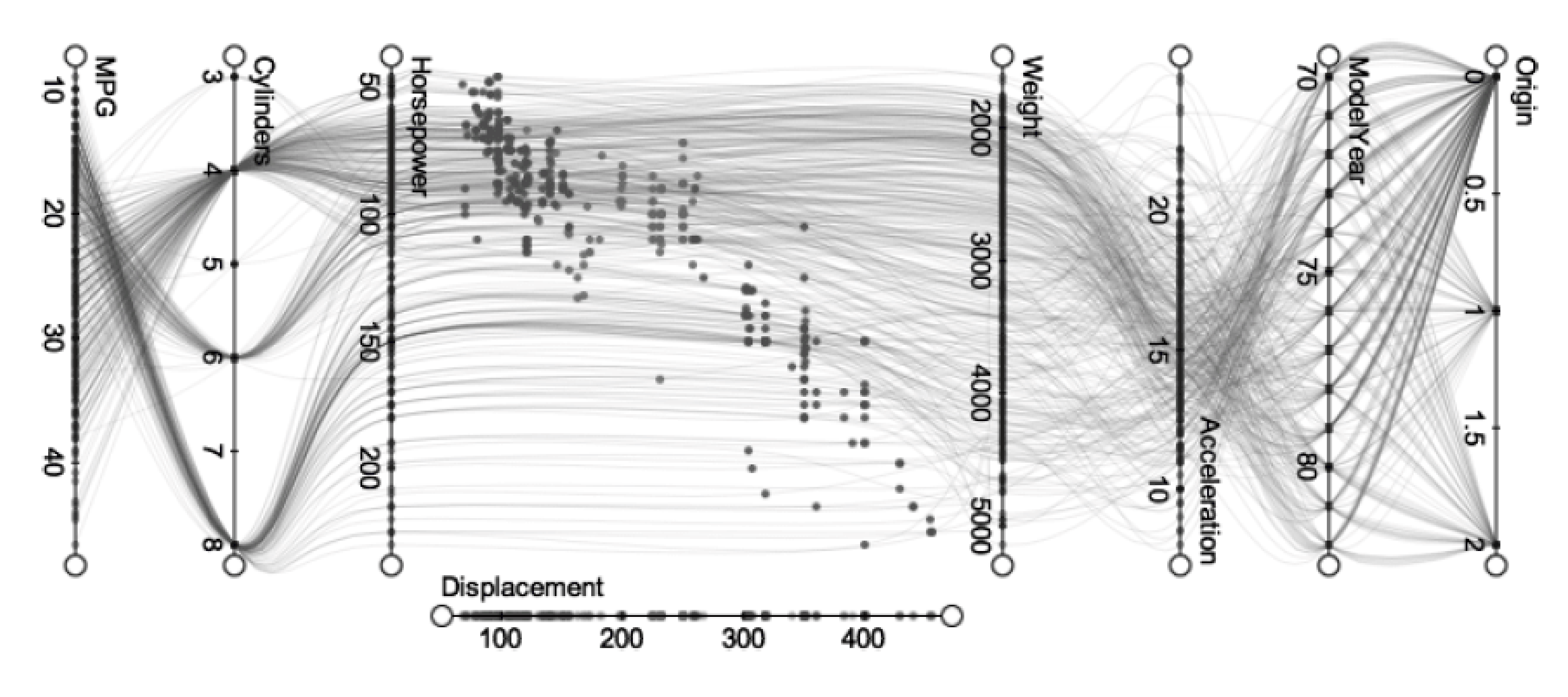


图 3-9　平行坐标结合散点图

图片来源：http://vis.pku.edu.cn/mddv/val/gallery

3.3　网络数据可视化

网络可视化充分利用视觉感知系统，将网络数据以图形化的形式展现出来，快速直观地解释及概览网络结构数据，一方面辅助用户认识网络的内部结构，另一方面有助于挖掘隐藏在网络内部的有价值信息。本节针对网络拓扑类型数据介绍几种对应的可视化方法，在 3.3.1 节中介绍常见的节点-链接方法，在 3.3.2 节中介绍相邻矩阵布局，在 3.3.3 节中介绍多种布局的混合应用。

互联网已经成为信息的主要载体之一，基于网络的研究在数据获取方面越来越容易，网络数据分析是伴随着现代网络技术的应用出现的一个较新的研究领域，它具有复杂性、多维度和时变性等特点。人们已经习惯在网络上分享信息，结识新的朋友等；基于微信、微博等社交 App，可以迅速发布身边正在发生的事情，因此社交媒体相对传统媒体（如报纸、电视新闻等）具有很好的竞争力。企业可以对巨大的网络数据进行挖掘分析并发现潜在的商业价值，甚至能通过基于网络的各种平台直接影响客户；客户同样可以从网络数据中获取信息来了解公司的方方面面，以达到指导和决定投资的目的。

相较于树形数据中明显的层次结构，网络数据并不具有自底向上或自顶向下的层次结构，表达的关系更加复杂和自由。图的绘制包括 3 个方面：网络布局、网络属性可视化和用户交互，其中最核心的要素是网络布局决定图的结构关系。最常用的网络布局方法有节点-链接法和相邻矩阵两类。

3.3.1　节点—链接法

在可视化布局中最自然的表达就是用节点表示对象，用线或者边表示关系的节点—链接布局（node-link）。它容易被用户理解、接受，帮助人们快速建立事物与事物之间的联系，显式地表达事物之间的关系，例如关系型数据库的模式表达、地铁线路图的表达，因而是网络数据可视化的首要选择。图 3-10 就是节点—链接法的一个例子。

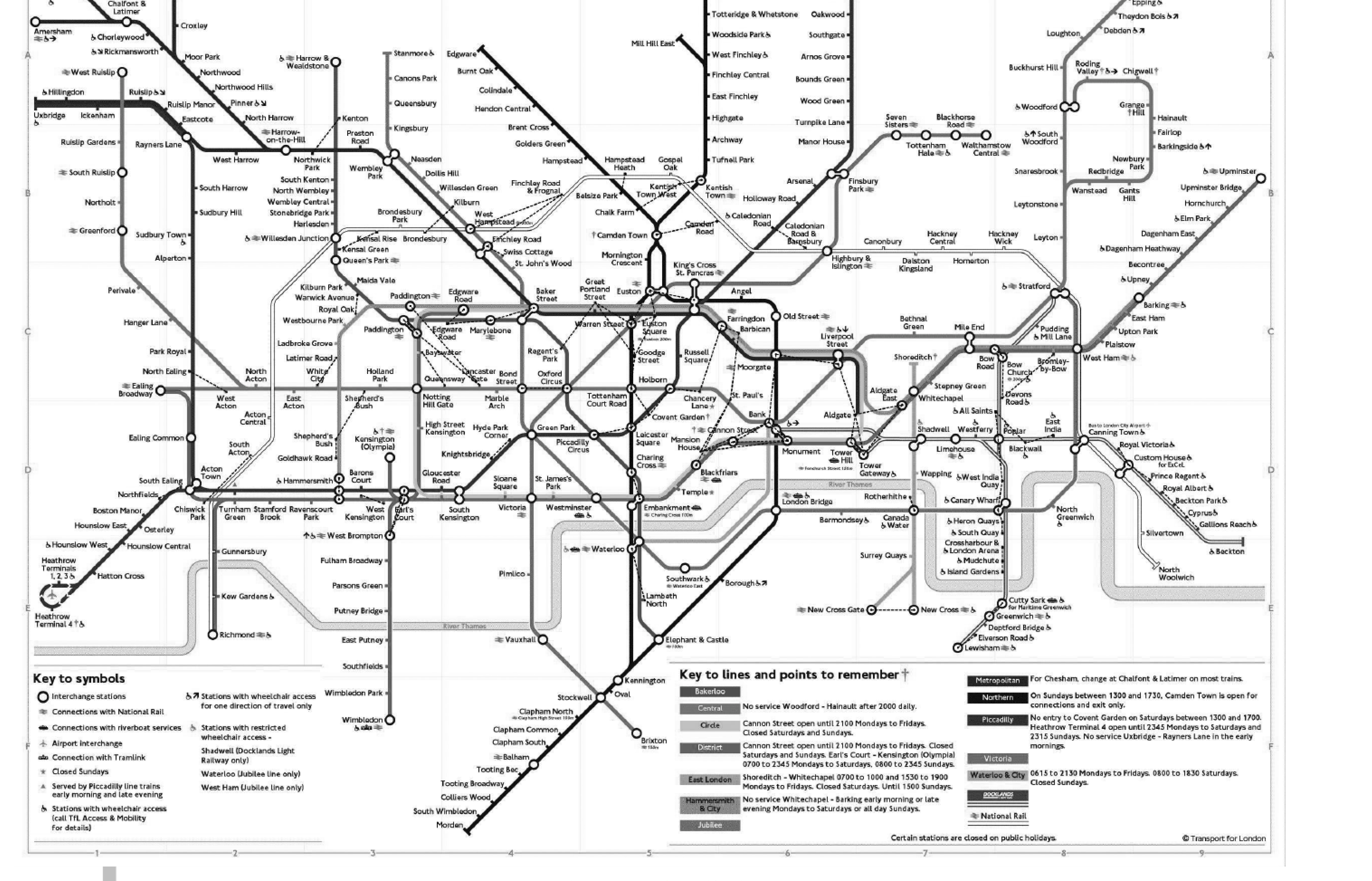

Key to symbols
Interchange stations
Connections with National Rail
Connections with riverboat services
Airport interchange
Connection with Tramlink
Closed Sundays
Served by Piccadilly line trains early morning and late evening
Stations with wheelchair access (call TfL Access & Mobility for details)
Stations with wheelchair access for one direction of travel only
Stations with restricted wheelchair access -
Shadwell (Docklands Light Railway only)
Waterloo (Jubilee line only)
West Ham (Jubilee line only)
Key to lines and points to remember
Bakerloo
Central
No service Woodford - Hainault after 2000 daily.
Circle
Cannon Street open until 2100 Mondays to Fridays. Closed Saturdays and Sundays.
District
Cannon Street open until 2100 Mondays to Fridays. Closed Saturdays and Sundays. Earl's Court - Kensington (Olympia) 0700 to 2345 Mondays to Saturdays, 0800 to 2345 Sundays.
East London
Shoreditch - Whitechapel 0700 to 1000 and 1530 to 1900 Mondays to Fridays. Closed Saturdays. Until 1500 Sundays.
Hammersmith & City
No service Whitechapel - Barking early morning or late evening Mondays to Saturdays or all day Sundays.
Jubilee
Metropolitan
For Chesham, change at Chalfont & Latimer on most trains.
Northern
On Sundays between 1300 and 1730, Camden Town is open for connections and exit only.
Piccadilly
No entry to Covent Garden on Saturdays between 1300 and 1700. Heathrow Terminal 4 open until 2345 Mondays to Saturdays and 2315 Sundays. No service Uxbridge - Rayners Lane in the early mornings.
Victoria
Waterloo & City
0615 to 2130 Mondays to Fridays. 0800 to 1830 Saturdays. Closed Sundays.
National Rail
Certain stations are closed on public holidays.
© Transport for London
River Thames
Amersham
Chalfont & Latimer
Chorleywood
Rickmansworth
Croxley
Moor Park
Northwood
Northwood Hills
Pinner
North Harrow
Harrow-on-the-Hill
Harrow & Wealdstone
West Ruislip
Hillingdon
Uxbridge
Ickenham
Ruislip
Ruislip Manor
Eastcote
Ruislip Gardens
Rayners Lane
South Ruislip
West Harrow
South Harrow
Northolt
Sudbury Hill
Greenford
Sudbury Town
Alperton
Perivale
Hanger Lane
Park Royal
North Ealing
Ealing Broadway
Ealing Common
South Ealing
Northfields
Boston Manor
Hounslow East
Osterley
Hounslow West
Hounslow Central
Heathrow Terminals 1, 2, 3
Hatton Cross
Heathrow Terminal 4
Kenton
Preston Road
Northwick Park
Wembley Park
South Kenton
North Wembley
Wembley Central
Stonebridge Park
Harlesden
Willesden Junction
Kensal Rise
Kensal Green
Queen's Park
Brondesbury Park
Brondesbury
Stanmore
Canons Park
Queensbury
Kingsbury
Edgware
Burnt Oak
Colindale
Hendon Central
Brent Cross
Golders Green
Hampstead
Neasden
Dollis Hill
Willesden Green
Kilburn
West Hampstead
Finchley Road & Frognal
Finchley Road
Swiss Cottage
St. John's Wood
Kilburn Park
Maida Vale
Warwick Avenue
Royal Oak
Westbourne Park
Ladbroke Grove
Latimer Road
White City
North Acton
West Acton
East Acton
Acton Central
South Acton
Acton Town
Chiswick Park
Turnham Green
Stamford Brook
Ravenscourt Park
Hammersmith
Goldhawk Road
Shepherd's Bush
Holland Park
Kensington (Olympia)
Barons Court
West Kensington
Earl's Court
West Brompton
Fulham Broadway
Parsons Green
Putney Bridge
East Putney
Southfields
Wimbledon Park
Wimbledon
Gunnersbury
Kew Gardens
Richmond
Paddington
Edgware Road
Marylebone
Baker Street
Bayswater
Queensway
Lancaster Gate
Marble Arch
Bond Street
Notting Hill Gate
High Street Kensington
Hyde Park Corner
Knightsbridge
Gloucester Road
South Kensington
Sloane Square
Victoria
St. James's Park
Westminster
Pimlico
Vauxhall
Stockwell
Clapham North
Clapham Common
Clapham South
Balham
Tooting Bec
Tooting Broadway
Colliers Wood
South Wimbledon
Morden
Brixton
Oval
Kennington
Elephant & Castle
Lambeth North
Waterloo
Southwark
Borough
Embankment
Temple
Blackfriars
Charing Cross
Leicester Square
Covent Garden
Piccadilly Circus
Green Park
Oxford Circus
Regent's Park
Great Portland Street
Warren Street
Euston Square
Euston
Goodge Street
Tottenham Court Road
Holborn
Russell Square
Chancery Lane
Mansion House
Cannon Street
St. Paul's
Bank
Monument
Tower Hill
Tower Gateway
London Bridge
Moorgate
Barbican
Farringdon
Angel
Old Street
Liverpool Street
Aldgate
Aldgate East
King's Cross St. Pancras
Mornington Crescent
Camden Town
Chalk Farm
Belsize Park
Hampstead Heath
Gospel Oak
Kentish Town West
Kentish Town
Camden Road
Caledonian Road
Caledonian Road & Barnsbury
Holloway Road
Arsenal
Highbury & Islington
Canonbury
Dalston Kingsland
Hackney Central
Homerton
Hackney Wick
Finsbury Park
Tufnell Park
Archway
Highgate
East Finchley
Finchley Central
West Finchley
Woodside Park
Totteridge & Whetstone
Mill Hill East
Oakwood
Southgate
Arnos Grove
Bounds Green
Wood Green
Turnpike Lane
Manor House
Seven Sisters
Tottenham Hale
Blackhorse Road
Walthamstow Central
Leyton
Leytonstone
Stratford
Snaresbrook
South Woodford
Woodford
Buckhurst Hill
Loughton
Debden
Theydon Bois
Epping
Roding Valley
Chigwell
Grange Hill
Hainault
Fairlop
Barkingside
Newbury Park
Redbridge
Wanstead
Gants Hill
Upminster
Upminster Bridge
Hornchurch
Elm Park
Dagenham East
Dagenham Heathway
Becontree
Upney
Barking
East Ham
Upton Park
Plaistow
West Ham
Bromley-by-Bow
Bow Church
Bow Road
Devons Road
Pudding Mill Lane
Mile End
Bethnal Green
Stepney Green
Whitechapel
Shoreditch
Shadwell
Wapping
Rotherhithe
Canada Water
Surrey Quays
Bermondsey
New Cross Gate
New Cross
Limehouse
Westferry
All Saints
Poplar
East India
Blackwall
West India Quay
Canary Wharf
Heron Quays
South Quay
Crossharbour & London Arena
Mudchute
Island Gardens
Cutty Sark
Greenwich
Deptford Bridge
Elverson Road
Lewisham
North Greenwich
Canning Town
Royal Victoria
Custom House
Prince Regent
Royal Albert
Beckton Park
Cyprus
Gallions Reach
Beckton
Silvertown
North Woolwich

图 3-10 伦敦地铁图

节点—链接法的优点是：比较直观地反映网络关系；能够表现图的总体结构、簇、路径；灵活，有许多的变种。节点—链接法的局限性有：几乎所有直观算法的复杂度均大于$O(N^2)$；对于密集(尤其是关系密集)的图不是很适用。

1. 力引导布局

力引导布局最早由 Peter Eades 在 1984 年的《启发式画图算法》一文中提出，目的是减少布局中边的交叉，尽量保持边的长度一致。此方法借用弹簧模型模拟布局过程：用弹簧模拟两个点之间的关系，受到弹力的作用后，过近的点会被弹开，而过远的点被拉近；通过不断的迭代，使整个布局达到动态平衡，趋于稳定。其后，“力引导”的概念被提出，演化成力引导布局算法(Fruchterman-Reingold 算法，简称 FR 算法)，即一种用于丰富两点之间关系的物理模型，加入点之间的静电力，通过计算系统的总能量并使得能量最小化，从而达到布局的目的。这种改进的能量模型可看成弹簧模型的一般化。

无论是弹簧模型还是能量模型，其算法的本质是要解决能量优化问题，区别在于优化函数的组成不同。优化对象包括引力和斥力部分，不同算法对引力和斥力的表达方式不同。对于平面上的两个节点 i 和 j，用 $d(i,j)$ 表示两个点的欧氏距离，$s(i,j)$ 表示弹簧的自然长度，k 是弹力系数，r 表示两个点之间的静电力常数，w 是两个点之间的权重。

弹簧模型为

$$E_s = \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{2}k(d(i,j)-s(i,j))^2$$

能量模型为

$$E = E_s + \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{rw_iw_j}{d(i,j)^2}$$

力引导布局易于理解和实现，可以用于大多数网络数据集，而且实现的效果具有较好的对称性和局部聚合性，因此比较美观。然而，力引导布局只能达到局部优化，而不能达到全局优化，并且初始位置对最后优化结果的影响较大。

算法 3-1 力引导布局的伪代码

```
设置节点的初始速度为(0,0)；设置节点的初始位置为任意但不重叠的位置。
1： 开始循环
2： 总动能：=0      //所有粒子的总动能为 0
3： 对于每个节点 i，净力 f：=(0,0)
4： 对于除该节点外的每个节点 j，净力 f：=净力 f+j 节点对应 i 节点的库仑斥力
5： 下一个节点 j+1
6： 对于该节点上的每个弹簧 s，净力 f：=净力 f+弹簧对该节点的胡克弹力
7： 下一个弹簧 s+1
8： //如果没有阻尼衰减，整个系统将一直运动下去
该节点速度：=(该节点速度+步长 * 净力) * 阻尼
该节点位置：=该节点位置+步长 * 该节点速度
总动能：=总动能+该节点质量 * (该节点速度)^2
9： 下一个节点 i+1
```

一般来说,整个算法的时间复杂度为 $O(n^3)$,迭代次数与步长有关,一般认为是 $O(n)$,每次迭代都要两两计算点之间的力和能量,复杂度是 $O(n^3)$。为了避免达到动态平衡后反复振荡,也可以在迭代后期将步长调到一个比较小的值。

例如,图 3-11 展示了一个小型的社交网络。社交网络是一个由个人或社区组成的点状网络拓扑结构。其中每个点(node)代表一个个体,可以是个人,也可以是一个团队或是一个社区,个体与个体之间可能存在各种相互依赖的社会关系,在拓扑网络中以点与点之间的边(tie)表示。而社交网络分析关心的正是点与边之间依存的社会关系。随着个体数量的增加以及个体间社会关系的复杂化,最后形成的整个社交网络结构可能会非常复杂。图 3-12 为法国作家维多克·雨果的小说《悲惨世界》的人物谱图。节点用颜色对通过子群划分算法计算的人物分类类别进行编码,边的粗细表示两个节点代表的人物之间共同出现的频率。

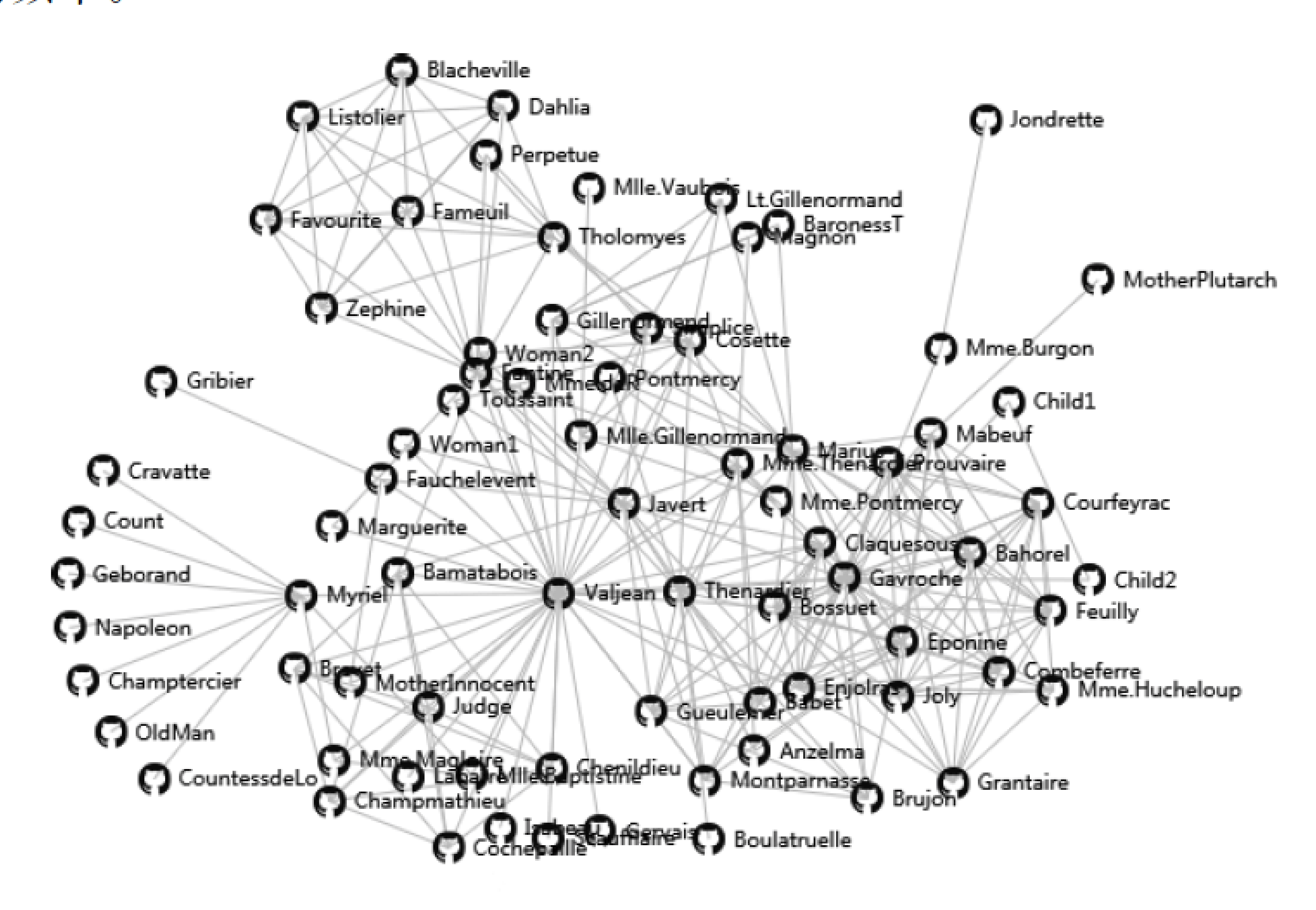


图 3-11 力引导布局-小型社交网络

图片来源:http://bl.ocks.org/mbostock/950642

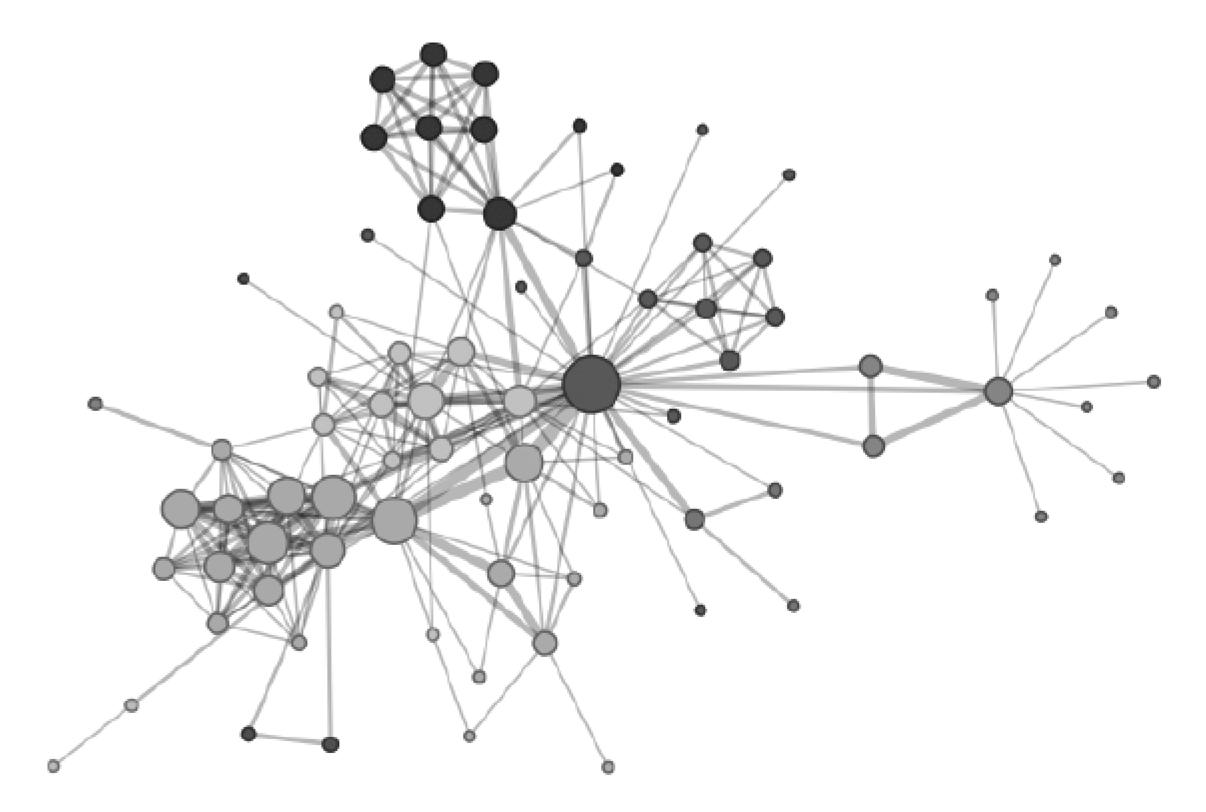


图 3-12 法国作家维多克·雨果的小说《悲惨世界》的人物谱图(力引导布局)

图片来源:http://homes.cs.washington.edu/~jheer///files/zoo/

2. 多维尺度分析布局

多维尺度分析(MDS)是一种探索性数据分析技术,主要是用适当的降维方法将多个变量通过坐标定位在低维空间(二维或三维)中,变量之间的欧氏距离就可以反映它们之间的差异性和相似性。MDS 布局的出现是为了弥补引导布局的局限性。它针对高维数据,用降维方法将数据从高维空间降到低维空间,力求保持数据之间的相对位置不变,同时也保持布局效果的美观性。力引导布局方法的局部优化使得在局部点与点之间的距离能够比较忠实地表达内部关系,但却难以保持局部与局部之间的关系。相对地,MDS 是一种全局控制,目标是要保持整体的偏离最小,这使得 MDS 的输出结果更加符合原始数据的特性。

MDS 根据数据集特征分为不考虑个体差异 MDS 模型和考虑个体差异 MDS 模型。MDS 模型允许多种类型的数据输入,并且在实际应用中,也有多种测量相似性或差异性的方法,根据分析数据的类型分为以下两种:

(1) 度量化 MDS 模型。也称为古典 MDS 模型,所输入的数据是直接反映变量间差异或相似的距离或比率,例如城市间的距离就是现成的反映差异的数据。

(2) 非度量化 MDS 模型。输入的数据不直接反映变量间的差异,而是通过对其属性的评分,间接地反映变量间的差异或相似性。

多维尺度分析的步骤如下:

(1) 界定问题。明确研究的问题和范畴,确定相关的变量种类和数量。

(2) 获取数据。根据实际情况获取分析数据。

(3) 选择 MDS 模型。根据获得的数据类型,选择相应的 MDS 模型。

(4) 确定维度。MDS 模型是为了生成一个用尽可能小的维度对数据进行最佳拟合的空间感知图,因此要确定一个合适的维度,维度太高不易于解读,维度太低会影响拟合度,通常采用二维或三维。

(5) 模型评价。考察应力系数 Stress 和拟合指数 RSQ,应力系数越小越好,拟合指数越大越好。

(6) 解读图表。多维尺度分析最重要的结果是感知图,图中各点之间的距离直接反映了各变量的相似或差异程度,除了查看差异程度之外,如果要对图表进行整体的分析解读,还需要对每个维度进行解释。

3. 弧长链接图

弧长链接图(arc diagram)是节点-链接法的一个变种。它将节点沿某个线性轴或环状排列,圆弧表达节点之间的链接关系。例如,图 3-13 是法国作家维多克・雨果的小说《悲惨世界》的人物谱图。图 3-14 是 2011 年年末欧债危机时 BBC 新闻制作的各国之间错综复杂的借贷关系的可视化。各个国家被放置在圆环布局上,曲线的箭头表示两国之间的债务关系,箭头的粗细表示债务的多少,颜色表示债务危机的严重程度,每个国家外债的数额决定了它们在圆环上所占的大小。由于债务关系太复杂,所以只有用户点击选中的国家的债务关系才被标注出来。欧洲最严重的是希腊,外债达到了国民生产总值的两倍。美国外债最多,主要的债权国是法国。弧长链接图不能像二维布局那样表达图的全局结构,但在节点良好排序后可清晰地呈现环和桥的结构。对节点的排序优化问题又称为序列化,在可视化、统计等领域有广泛的应用。

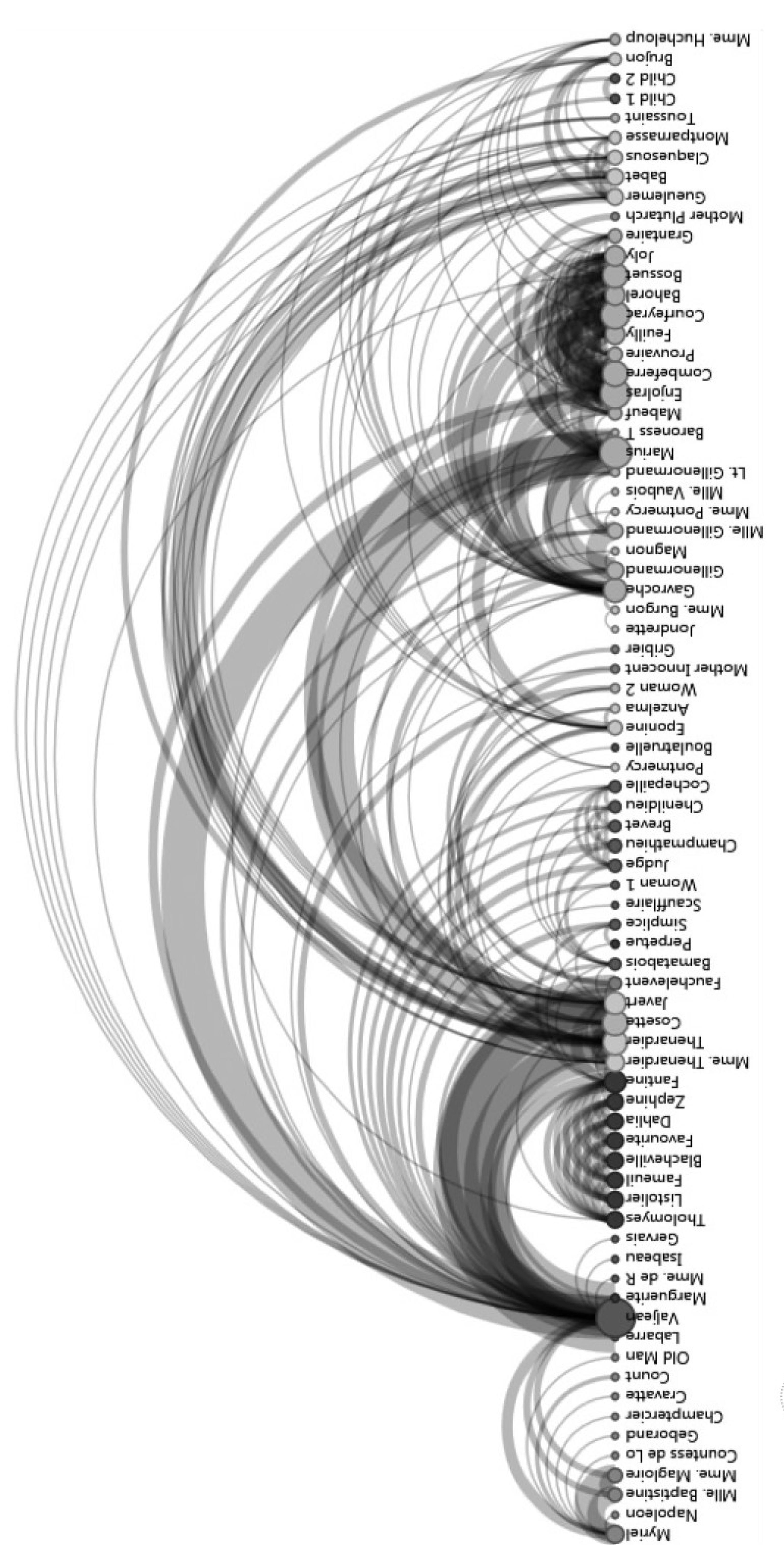


图 3-13 法国作家维多克·雨果的小说《悲惨世界》的人物谱图（弧长链接图）

图片来源：http://homes.cs.washington.edu/~jheer//files/zoo/

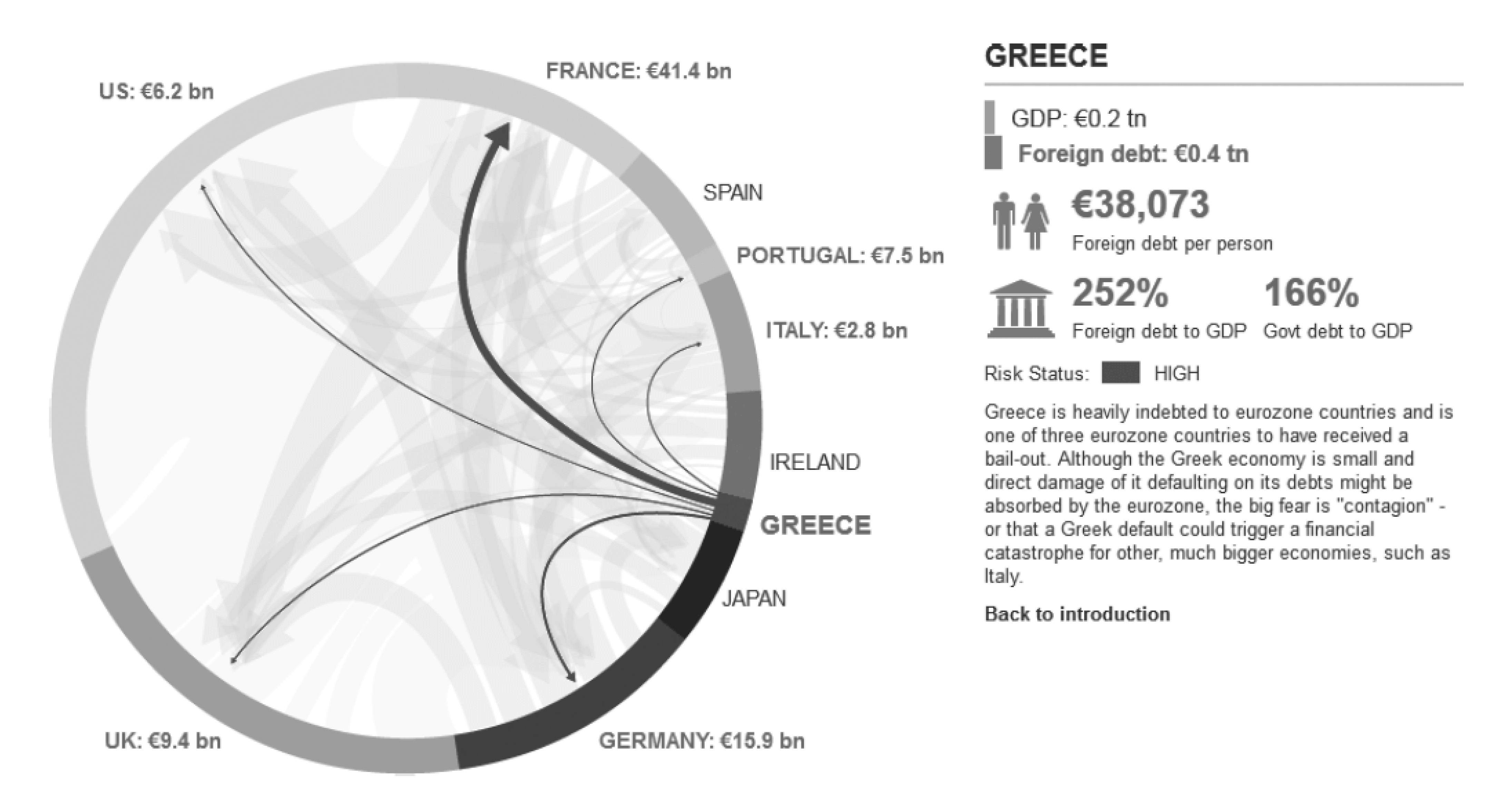


图 3-14　2011 年年末欧债危机时 BBC 新闻制作的各国之间错综复杂的借贷关系的可视化

图片来源：http：//www. bbc. co. uk/news/business-15748696

3.3.2　邻接矩阵布局

邻接矩阵(adjacency matrix)指代表 N 个节点之间关系的 $N \times N$ 的矩阵，矩阵内的位置(i,j)表达了第 i 个节点和第 j 个节点之间的关系。对于无权重的关系网络，用 0-1 矩阵(binary matrix)来表达两个节点之间的关系是否存在；对于带权重的关系网络，邻接矩阵则可用(i,j)位置上的值代表其关系紧密程度；对于无向关系网络，邻接矩阵是一个对角线对称矩阵；对于有向关系网络，邻接矩阵的对角线表达节点与自己的关系。邻接矩阵关系图的示例如图 3-15 所示。

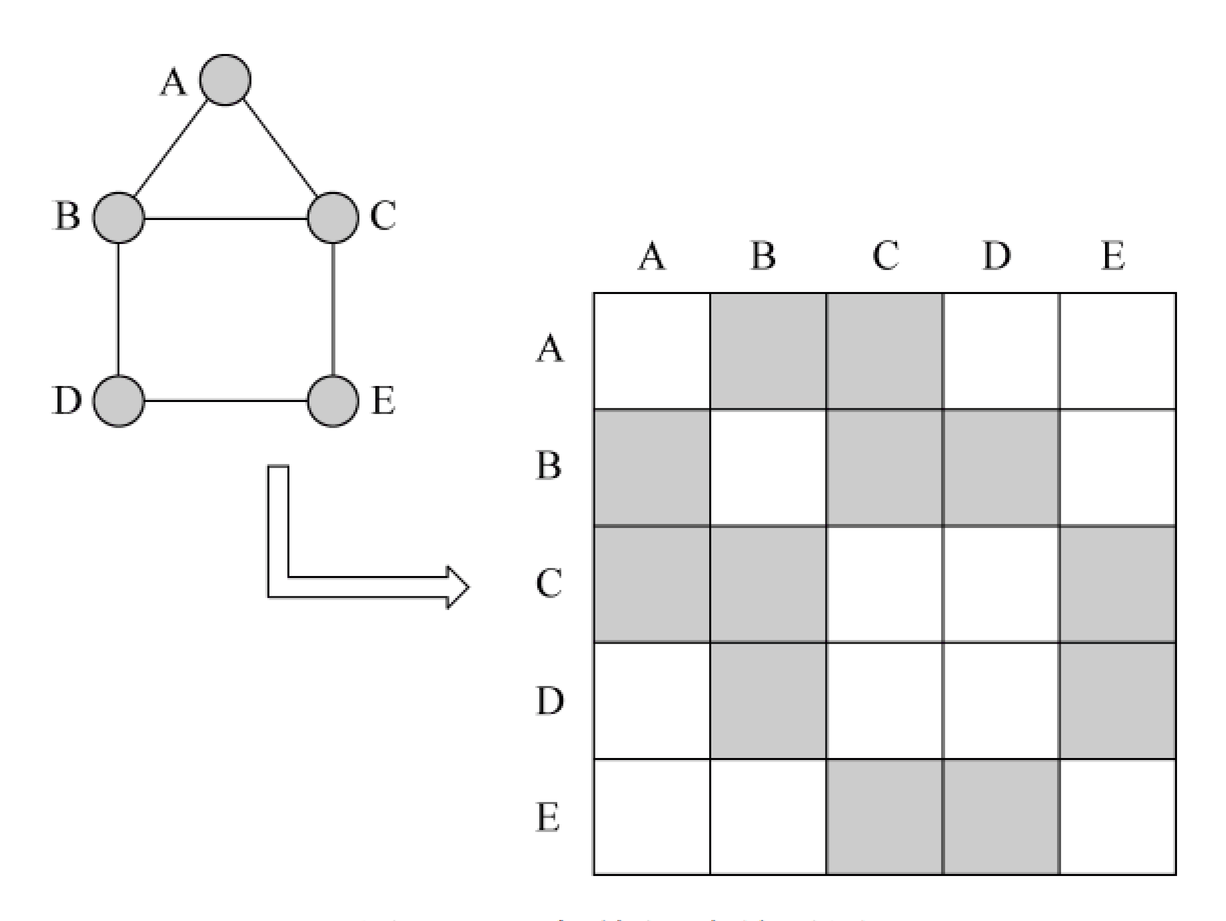


图 3-15　邻接矩阵关系图

邻接矩阵布局能够完全规避边的交叉，所以非常适用于密集的图。其视觉伸缩性强；

能更好地表达两两关联的网络数据。但同时它的可视化结果比较抽象，难以呈现网络的拓扑结构。另外，它不能直观地表达关系传递性和网络中心性，难以跟踪出路径。

邻接矩阵的表达简单易用，可以用数值矩阵，也可以将数值映射到色彩空间表达。但是，从邻接矩阵中挖掘出隐藏的信息并不容易，需要结合人机交互。最关键的两种交互是排序和路径搜索，前者使具有相似模式的节点靠得更近，而后者则用于探索节点之间的传递关系。图 3-16 为邻接矩阵法实例，是城市交易邻接矩阵，横向和纵向分别表示收货和发货的关系。饱和度越低，颜色越白，表示交易值越小；颜色越深，表示交易值越大。

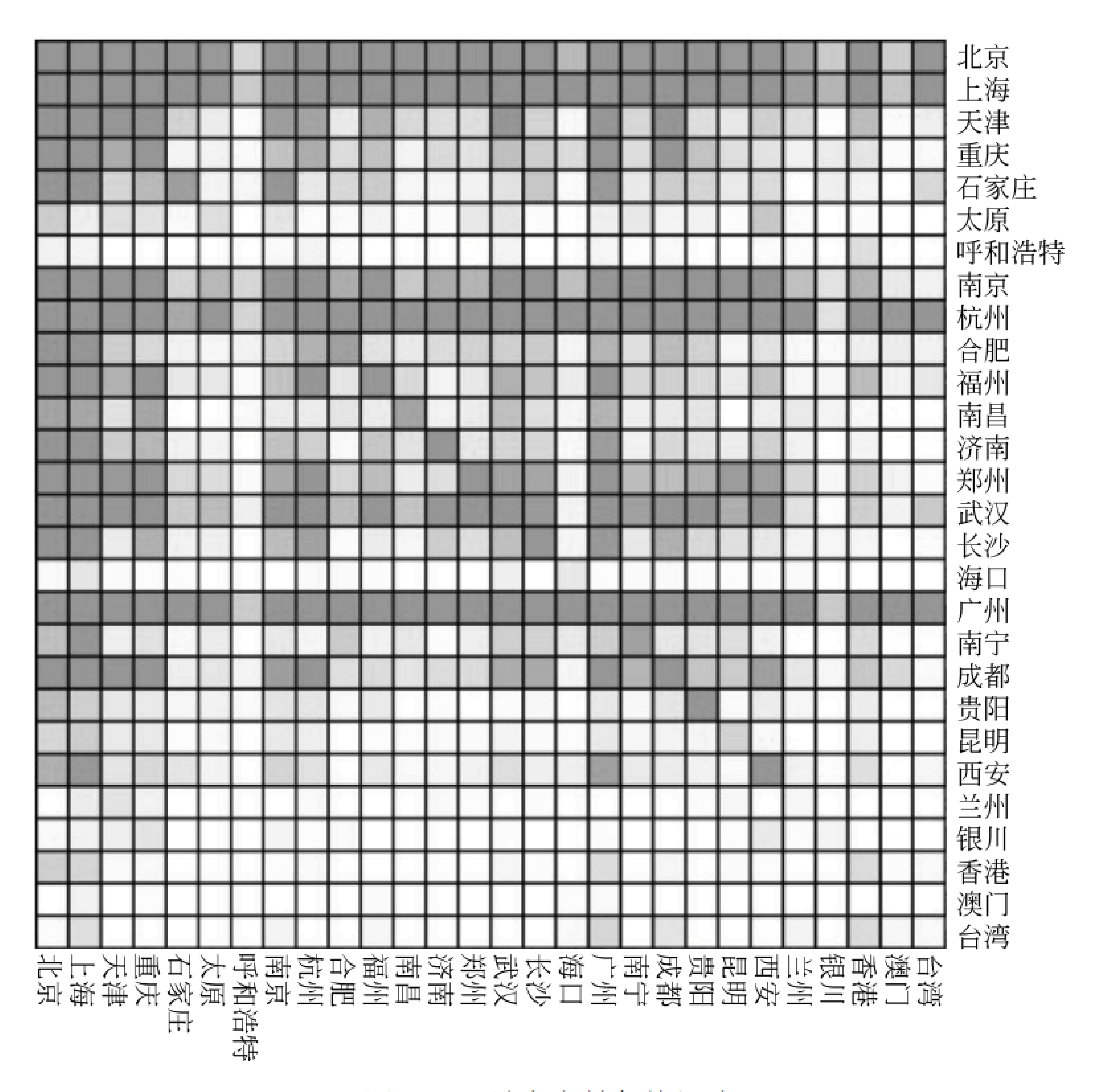


图 3-16　城市交易邻接矩阵

邻接矩阵排序的意义是凸显网络关系中存在的模式。类似于弧长链接图，这个问题也称为序列化问题。一个 $N \times N$ 的邻接矩阵共有 $n!$ 种排序方式，在这 $n!$ 种组合中找到使代价函数最小的排序方式称为最小化线性排列，是一个 N-P 难度的问题。在实际应用中，通常采用启发式算法，不求达到最优。

常规的排序方法依据网络数据的某一数值（矩阵值或节点的度）的大小执行。选择不同的排序项产生不同的矩阵排序结果。图 3-17 为法国作家维多克・雨果的小说《悲惨世界》的人物谱图，图中采用子群聚类算法获得的人物分类结果对邻接矩阵的行和列进行排序。

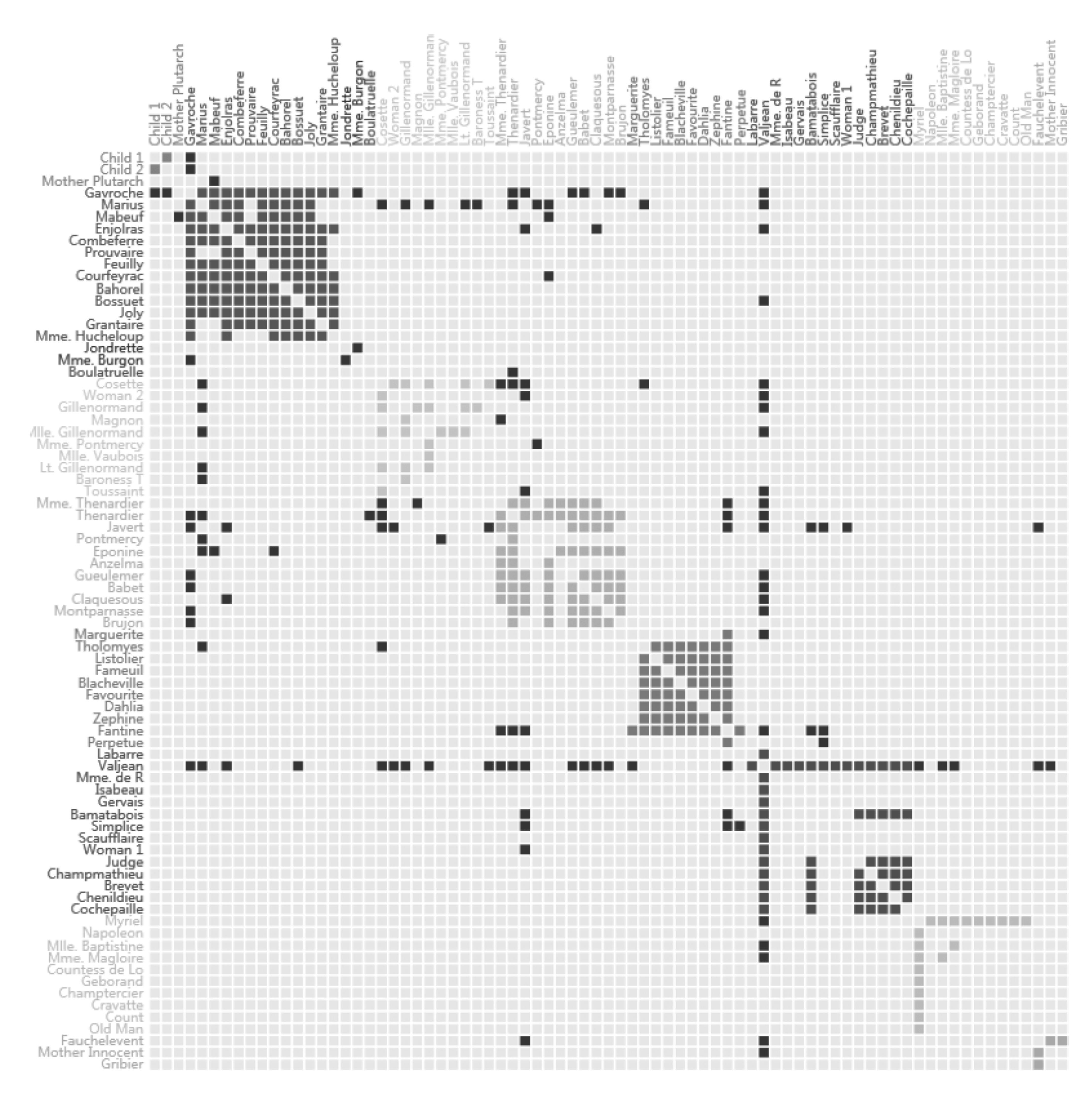


图 3-17　法国作家维多克·雨果的小说《悲惨世界》的人物邻接矩阵

图片来源：http：//homes. cs. washington. edu/～jheer///files/zoo/

3.3.3　混合布局

节点—链接布局适用于节点规模大但边关系较为简单，并且能从布局中看出图的拓扑结构的网络数据；而邻接矩阵恰恰相反，适用于节点规模较小，但边关系复杂，甚至是两两节点之间都存在关系的数据。这两种数据的特点是用户选择布局的首要区分原则。对于部分稀疏、部分稠密的数据，单独采用任何一种布局都不能良好地表达数据，这时便可以混合两者的布局来设计。一些人认为不同的布局各有所长，取长补短的混合布局永远是优于单一布局的选择，但实际上顾此失彼的例子常常见到。是否需要混合布局、如何设计混合布局必须经过仔细思考。

是否一种布局已经足以表达数据的模式？无须为了混合而混合，尤其不应为了花哨的可视效果而混合。可视化是一个科学严谨的过程，如果增加布局并不能表达重要的额外信息特征，或者另一种布局的可视化效果重叠，那么混合布局不是一种好的选择。

多种布局如何混合成一种布局？简单的多种布局罗列称为仪表盘(dashboard)或多视图模式，即将不同的布局结果放置于同一个页面，在视图之间实现可视交互的同步。根据可视化布局组合研究，除并列式组合外还有载入式、嵌套式、主从式、结合式4种组合模式，对可视化布局更丰富的组合能大大提高可视化结果的分析效率。

3.4 可视化案例分析

在本节中将结合两个实际案例来介绍可视分析的实际应用。在3.4.1节中介绍在完成China VIS 2015比赛中所采用的一些可视化方法以及分析过程。在3.4.2节中介绍在完成VAST Challenge 2016比赛中所采用的一些可视化方法以及分析过程。

3.4.1 案例一：China VIS 2015竞赛题

1. 简介

本题关注对正常网络流量日志数据的分析。TSZNet公司提供了一周的tcpflow日志数据，希望参赛者找到公司内部网络的客户端和服务器，分析公司内网的网络体系结构，总结公司内网的正常网络通信模式有哪些。

2. 数据

一周的tcpflow(TCP协议层的数据传输记录)日志，正常流量。tcpflow日志字段包括time(时间)、sip(源IP)、dip(目的IP)、proto(协议)、dport(目的端口)、uplen(上行字节数)、downlen(下行字节数)

3. 问题

(1) 找出TSZNet公司内部网络中的客户端与服务器，并给出该公司网络的体系结构拓扑图。

(2) 对TSZNet公司内部网络中的服务器进行分类，分类标准不限，比如按照功能、时间特点、行为特点、流量特点等。

(3) 说明TSZNet公司内部网络的客户端-服务器，客户端-客户端，服务器-服务器之间的常规通信模式。

4. 解答

(1) 通过对一周tcpflow数据的可视分析，找出TSZNet公司内部网络中的客户端与服务器，并给出该公司网络体系结构拓扑图。

首先，根据日志记录中内网与内网之间的通信数据，用力导向图来表示内网之间的通

信(图 3-18),每一个节点代表一个主机。用连线表示两台主机之间有通信,连线的宽度代表两个节点之间的流量大小,其中颜色表示使用最多的通信协议,节点的大小表示流量。从图 3-18 中可以看出,有一些主机与很多台主机都有连接关系,并且流量大于大多数主机,那么可以认为该主机可能为服务器。然后依次找出可能是服务器的主机,如图中黑框内所示,然后进行下一次筛选。

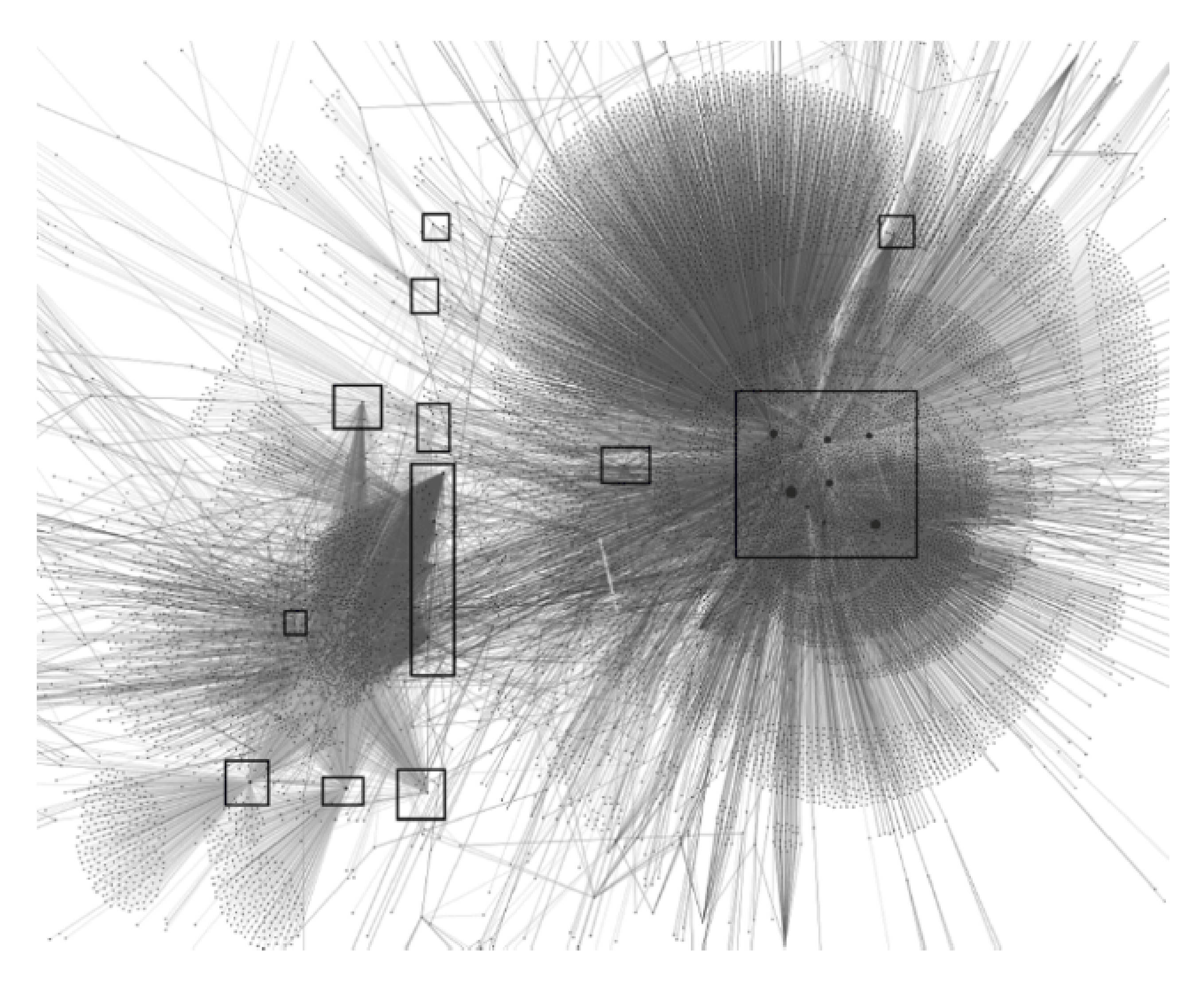


图 3-18　内网通信的力导向图

其次,根据第一次筛选结果再次画出通信网络图(图 3-19),然后从中去除孤立的点和度小于平均度的节点。剩下的基本可以确定为服务器。

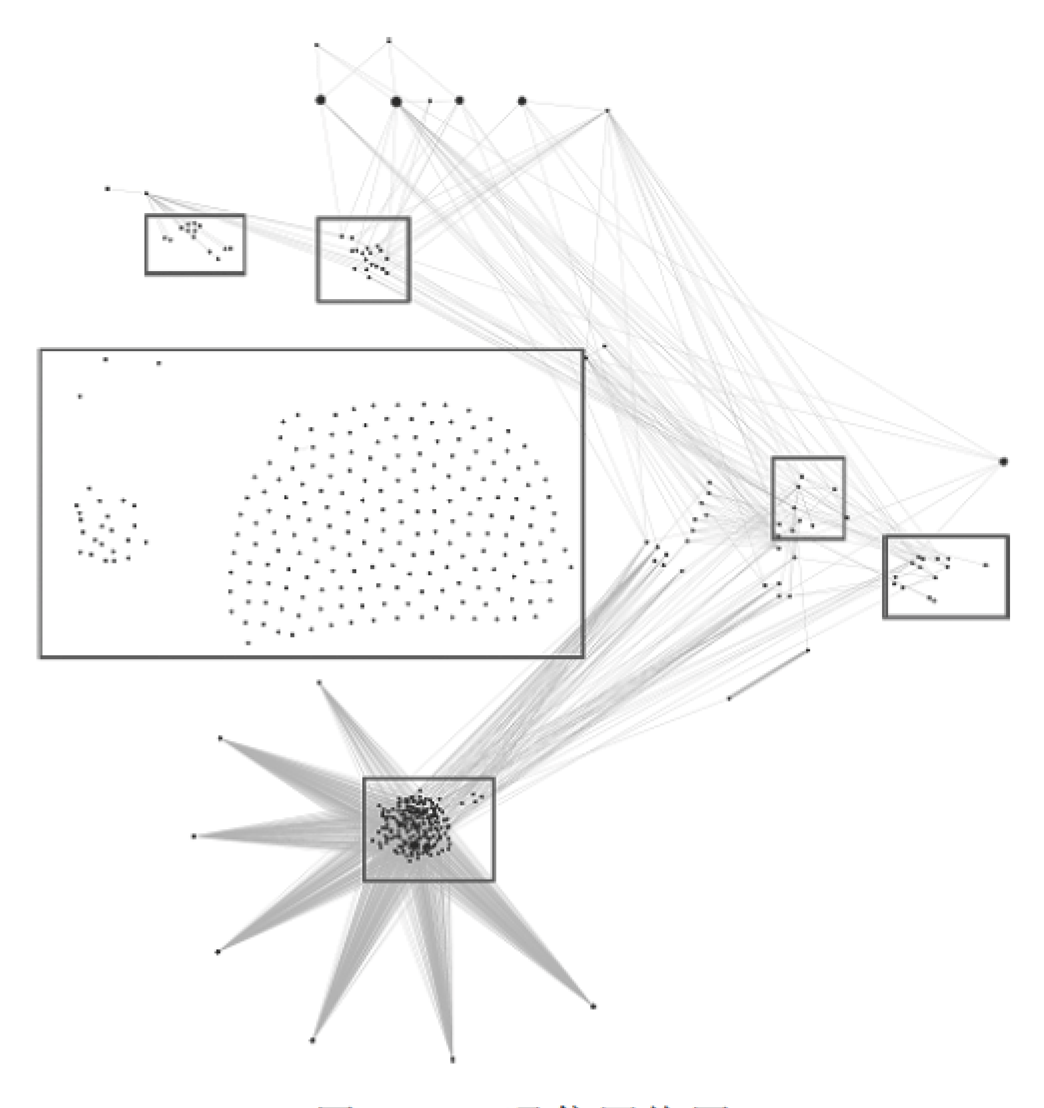


图 3-19　通信网络图

图 3-20 表示内网主机与服务器之间的通信关系,其中白色的节点表示客户端的 IP

段,精确到每个 IP 地址的前 3 段,如 10.59.23.×。红色的代表服务器,同样,服务器流量用节点大小编码。

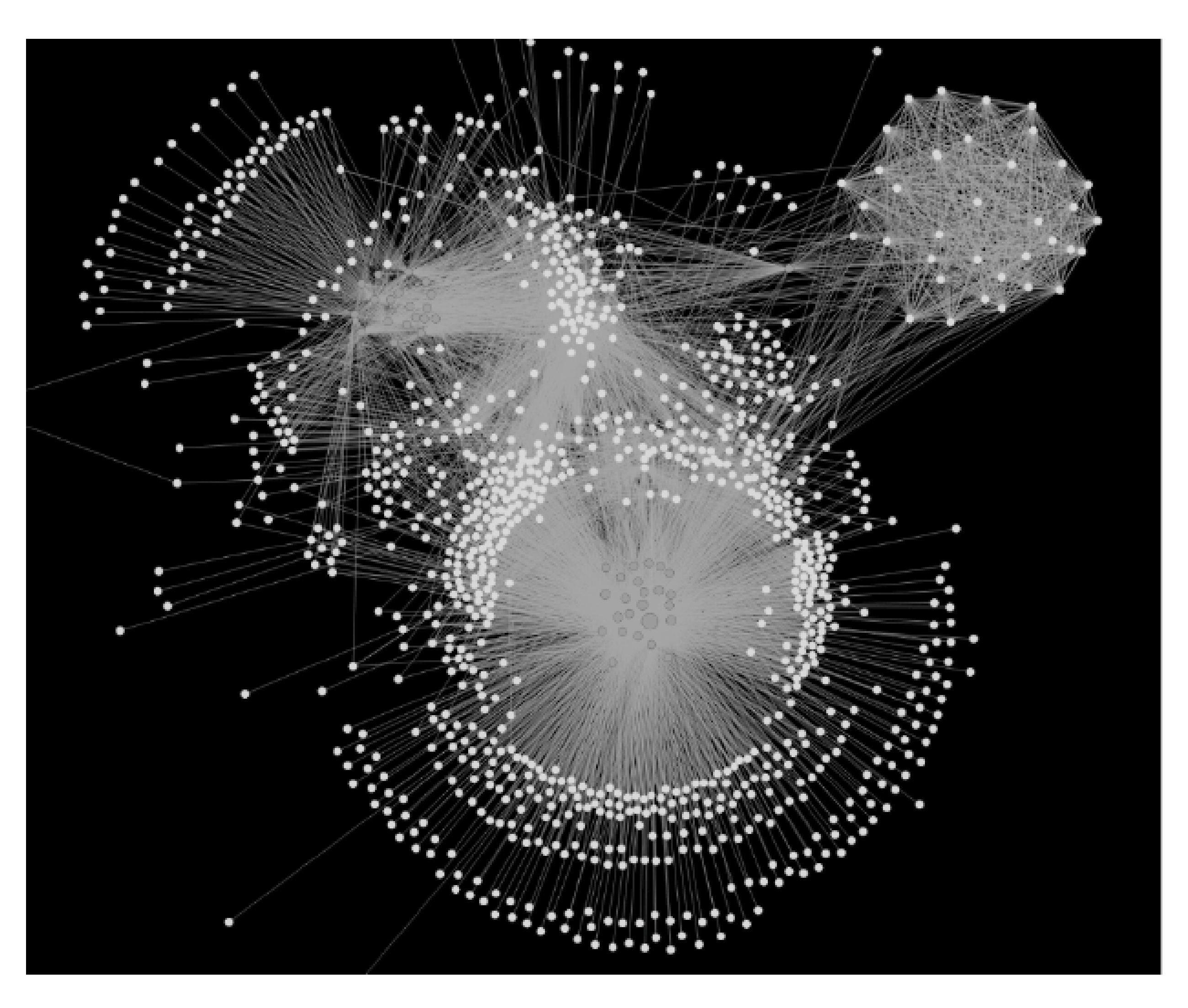


图 3-20 内网主机与服务器之间通信关系的力导向图

图 3-21 为公司网络体系结构拓扑图。从通信日志可以看出,公司内部使用 A 类私有地址,因此可以根据客户端 IP 所在的 IP 段来对客户端进行分组,每个网段内的主机通过核心交换机与内网中其他网段主机和服务器进行通信。服务器与核心交换机直接相连,内网主机通过代理服务器访问外网。

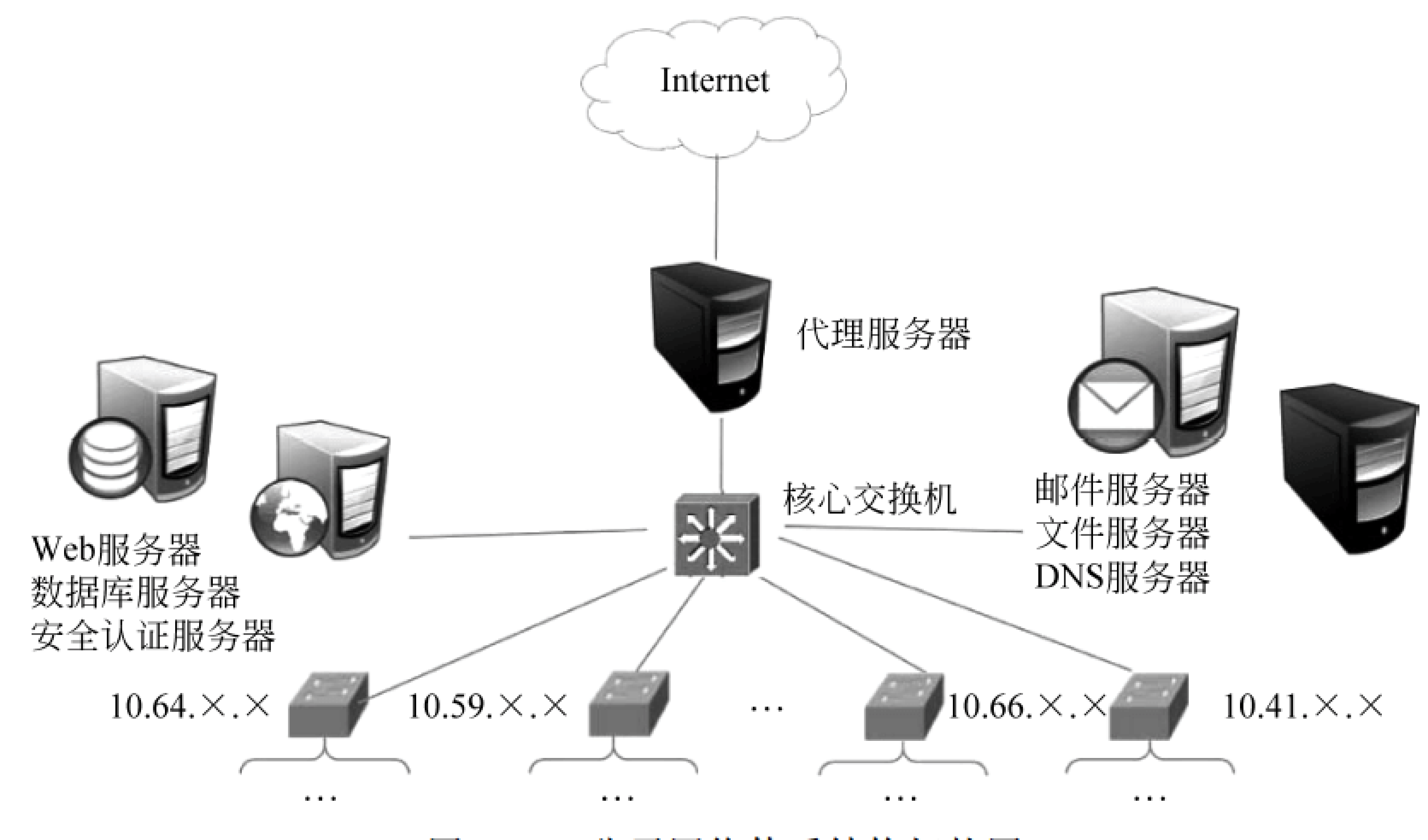


图 3-21 公司网络体系结构拓扑图

(2)通过对一周的 tcpflow 数据的可视分析,对 TSZNet 公司内部网络的服务器进行分类,分类标准不限,比如按照功能、时间特点、行为特点、流量特点等。

主要通过功能来对服务器进行分类,因此主要通过日志记录中的协议来区分每台服

务器的功能。首先使用 Python 统计出每台服务器通过什么样的协议与多少台不同的客户端主机通信,然后根据这些统计数据做分析。

根据统计结果,使用了热点图的可视化方式来从中发现每台服务器的类型(图 3-22)。在图 3-22 中,横轴表示每台服务器的 IP,纵轴表示服务器使用的协议,然后可以画出热点图,其中每个矩形都对应一台服务器和协议,颜色表示对应的服务器通过对应的协议通信的客户端的数量。颜色越深,代表与之通信的客户端数量越多;而颜色浅的代表对应的服务器通过对应的协议与客户端通信数量比较少甚至没有。从图 3-22 中可以看出,HTTP 协议使用最多,那么可以推断出这些服务器中大都是 Web 服务器。

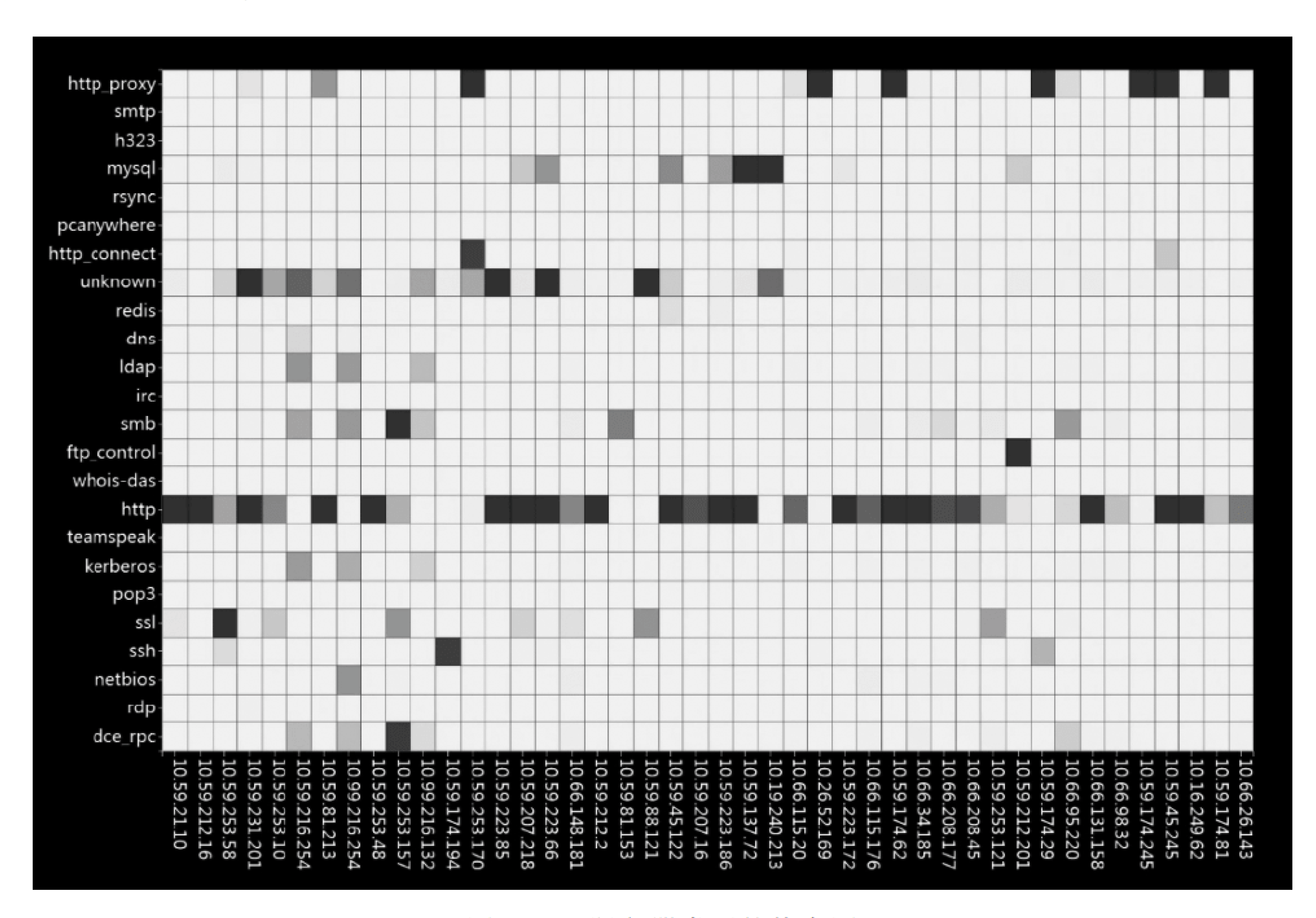


图 3-22　服务器类型的热点图

此外,可以通过交互的方式来查看某台服务器通过某种协议与多少台客户端进行通信(图 3-23)。对于一台服务器而言,主要通过判断颜色的深浅来确定这台服务器主要使用的协议。在图 3-23 中,查看了 10.59.223.186 这台服务器通过 HTTP 协议与多少台客户端通信,鼠标移动到对应的矩形上的时候,就会显示对应的矩形代表的信息。如图所示,10.59.223.186 这台服务器通过 HTTP 协议与超过 10 000 台客户端通信(大于 10 000 的记为 10 000),那么可以确定 10.59.223.186 这台服务器的类型为 Web 服务器。

最后,图 3-24 表示对服务器进行分类的结果,使用分区图来表示,最上面一层是根节点,表示下面的都是服务器,第二层和第三层分别表示服务器的类型和服务器的 IP,颜色相同表示的是同一种类型的服务器。从图中可以看出服务器总共分为 8 类。有 Web 服务器、代理服务器、dns_ldap 服务器、文件服务器、邮件服务器和数据库服务器等。

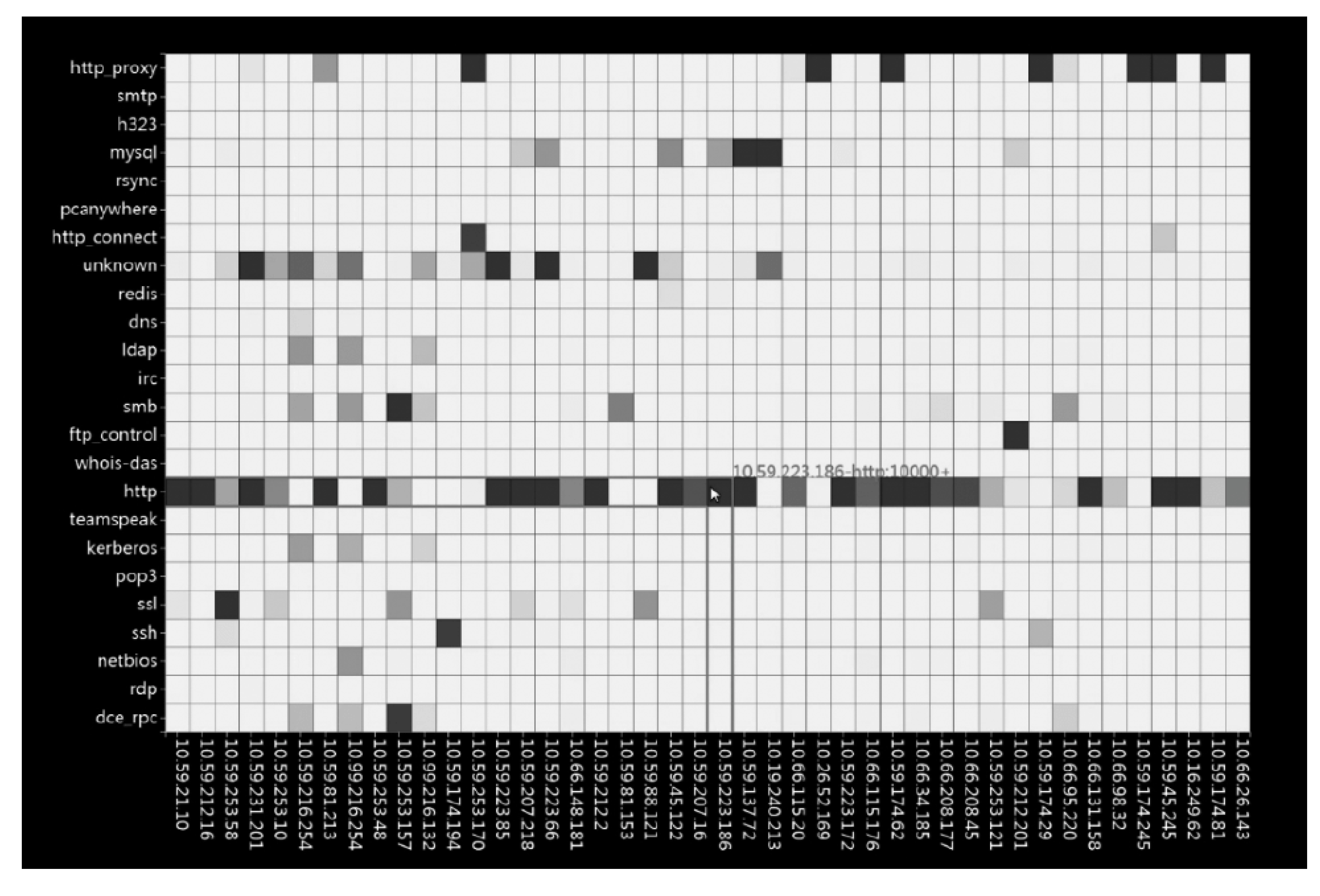


图 3-23　服务器与客户端通信的热点图

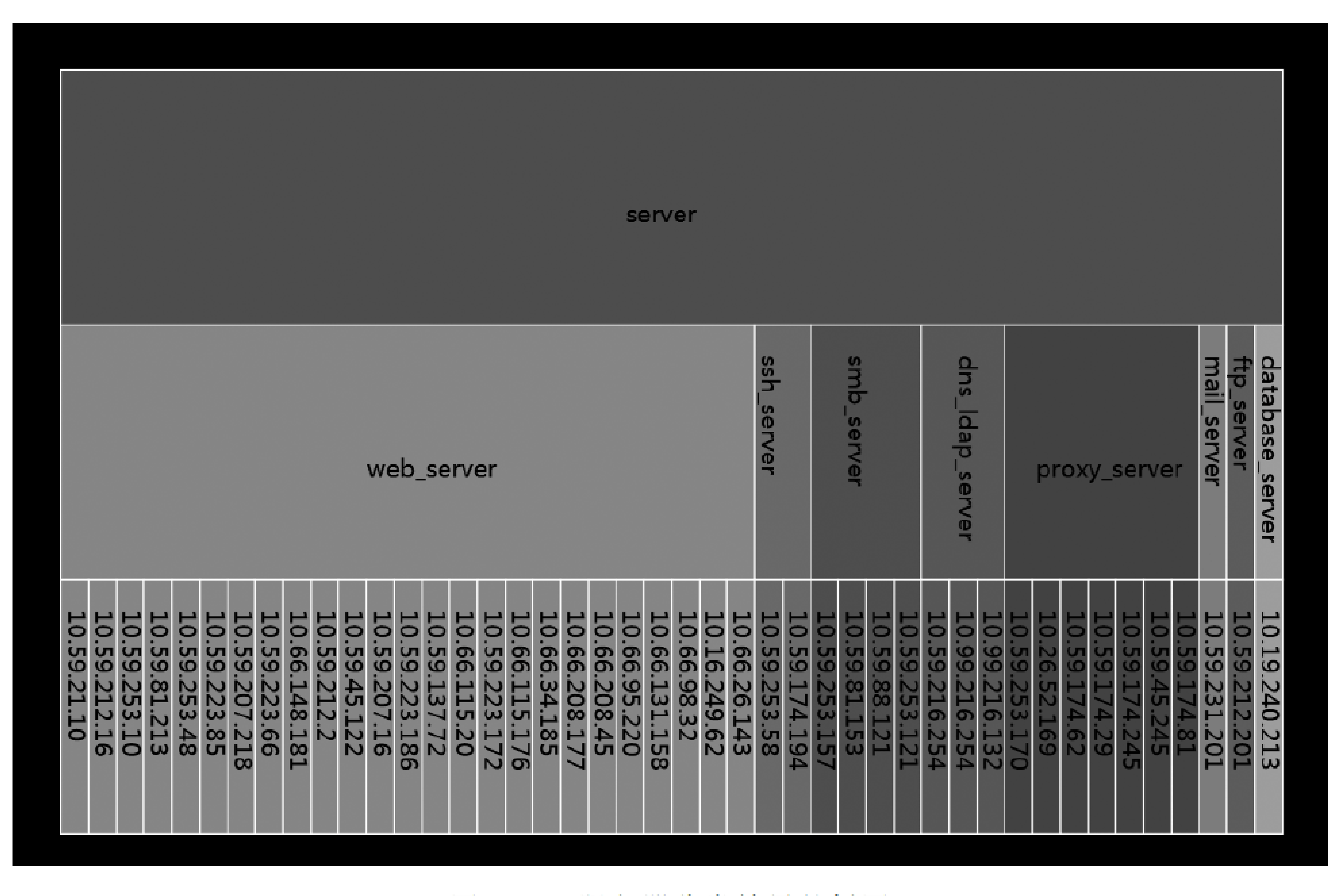


图 3-24　服务器分类结果的树图

(3) 通过对一周的 tcpflow 数据的可视分析,说明 TSZNet 公司内部网络的客户端-服务器、客户端-客户端、服务器-服务器之间的常规通信模式。

为了发现客户端与服务器之间的通信模式,根据内部通信数据绘制平行坐标系,如图 3-25 所示。因为客户端数量比较多,不适合使用 D3. js 同时绘制,所以使用 IP 段来表示,图中左边坐标轴表示客户端 IP 段,中间表示使用的协议,右边表示服务器 IP。然后根据数据,将有通信关系的客户端和服务器通过使用的协议连接起来。

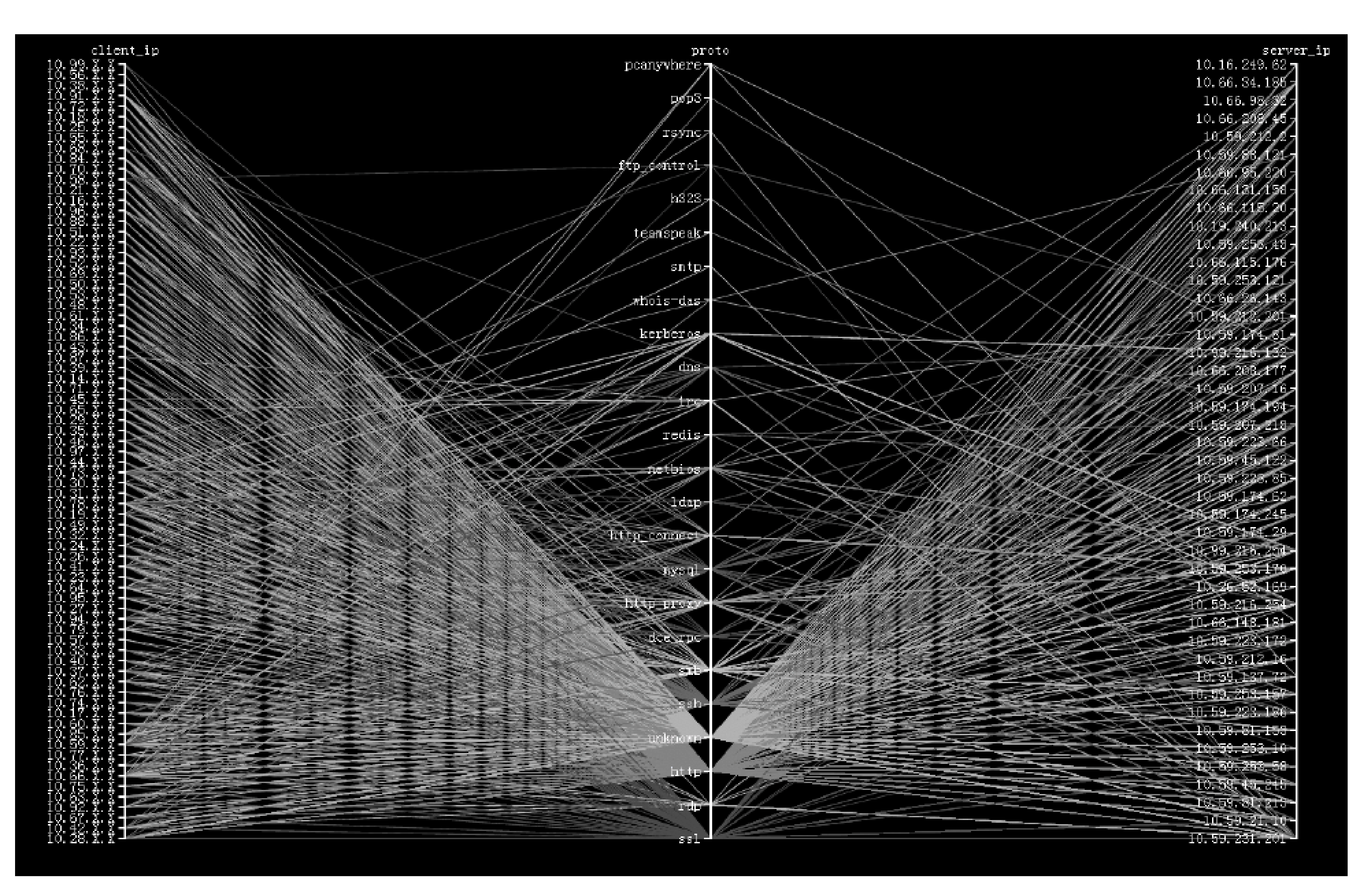


图 3-25　内部通信的平行坐标系

从图中可以看出,在客户端与服务器的常规通信模式中,使用 http 协议和 unknown 协议居多,http 为常规的 web 通信协议,unknown 协议猜测可能为公司内部开发的通信协议。此外 ssl 和 ssh 协议也使用比较多,表示客户端和服务器通过比较安全的方式进行通信。其他使用比较多的通信协议还有 http_proxy、mysql、smb 等。

在图 3-25 中可以看出整体通信模式,当然也可以查看指定的客户端 IP 段的主要的通信模式,如图 3-26 所示,可以选择查看特定的客户端通过什么样的协议与哪些服务器通信。例如,10. 26. ×. ×这个网段内的主机主要通过 http、unknown 等协议与服务器进行通信。

服务器与服务器之间通信模式使用如图 3-27 所示的弧长链接图来表示。从图中可以看出,服务器与服务器之间主要通过 http、ldap、dns、dec_rpc 等协议进行通信。

客户端与客户端之间的通信模式同样使用平行坐标的方式来展示,如图 3-28 所示,左右两端分别表示网段,因为主机数量比较多,所以没有将所有主机一一列出。从图中可以看出,客户端与客户端之间主要通过 unknown、http 等协议通信,此外还有 ssl、ssh 等协议。

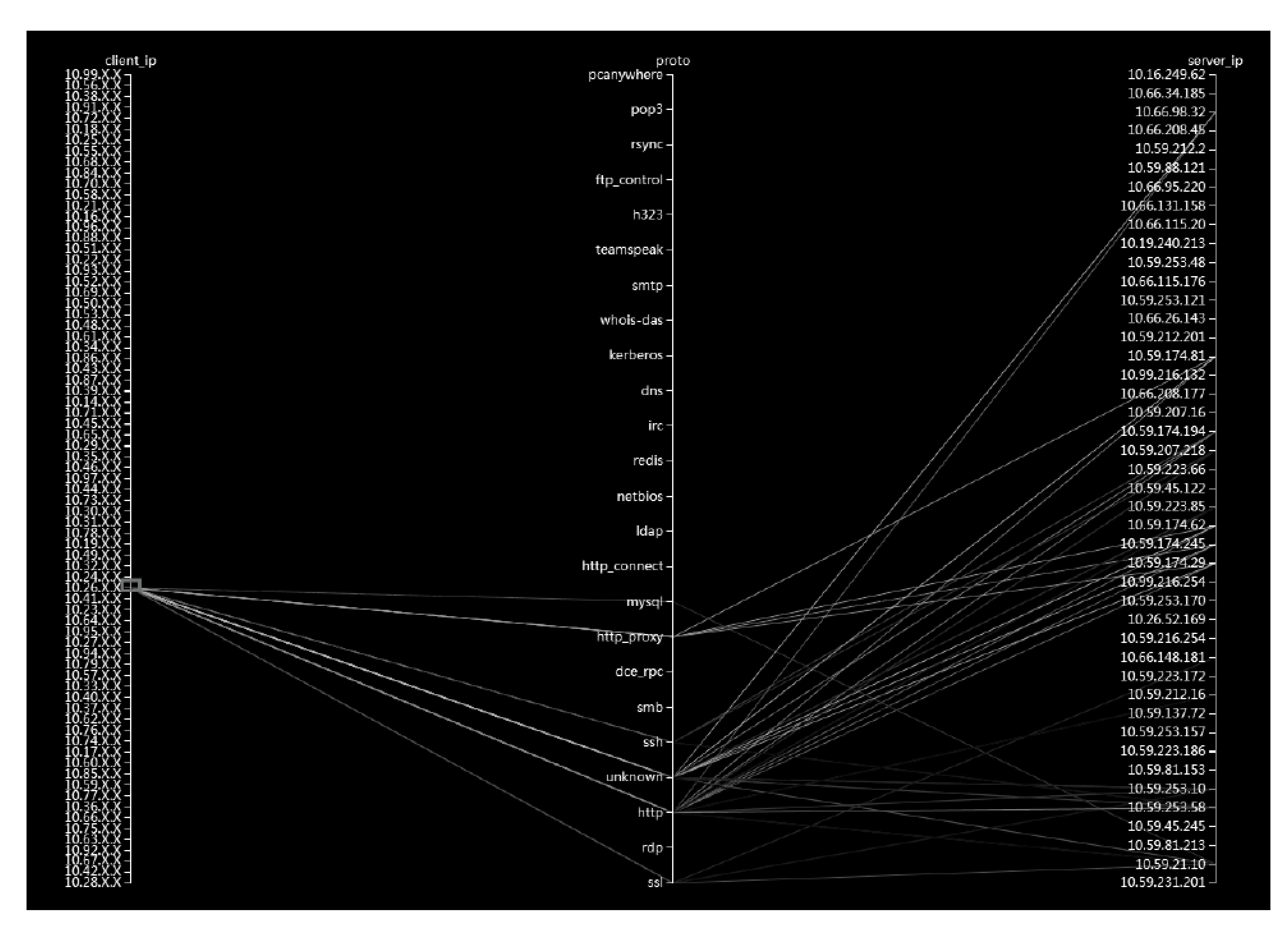


图 3-26　内部通信的平行坐标系

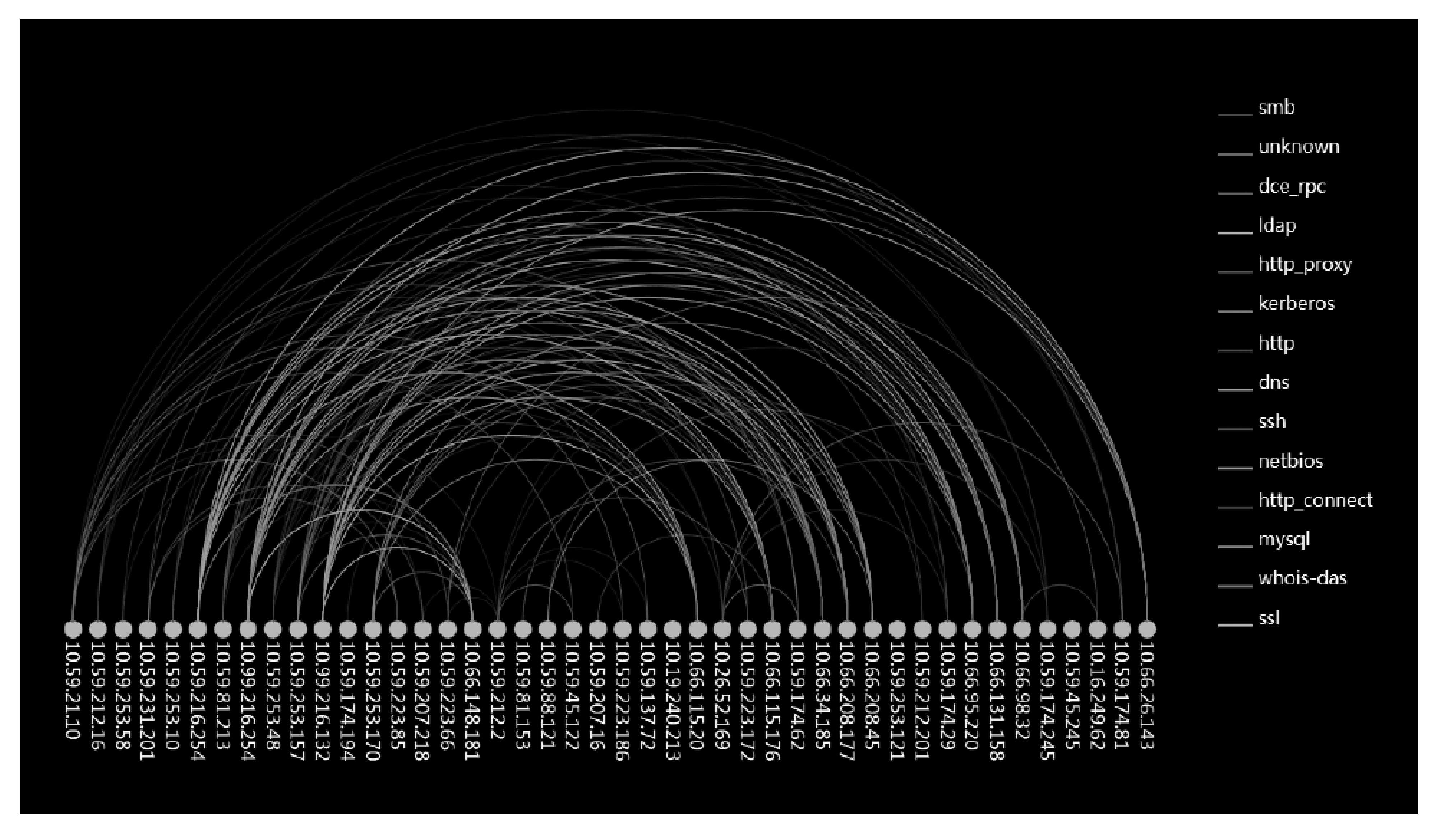


图 3-27　服务器与服务器通信的弧长链接图

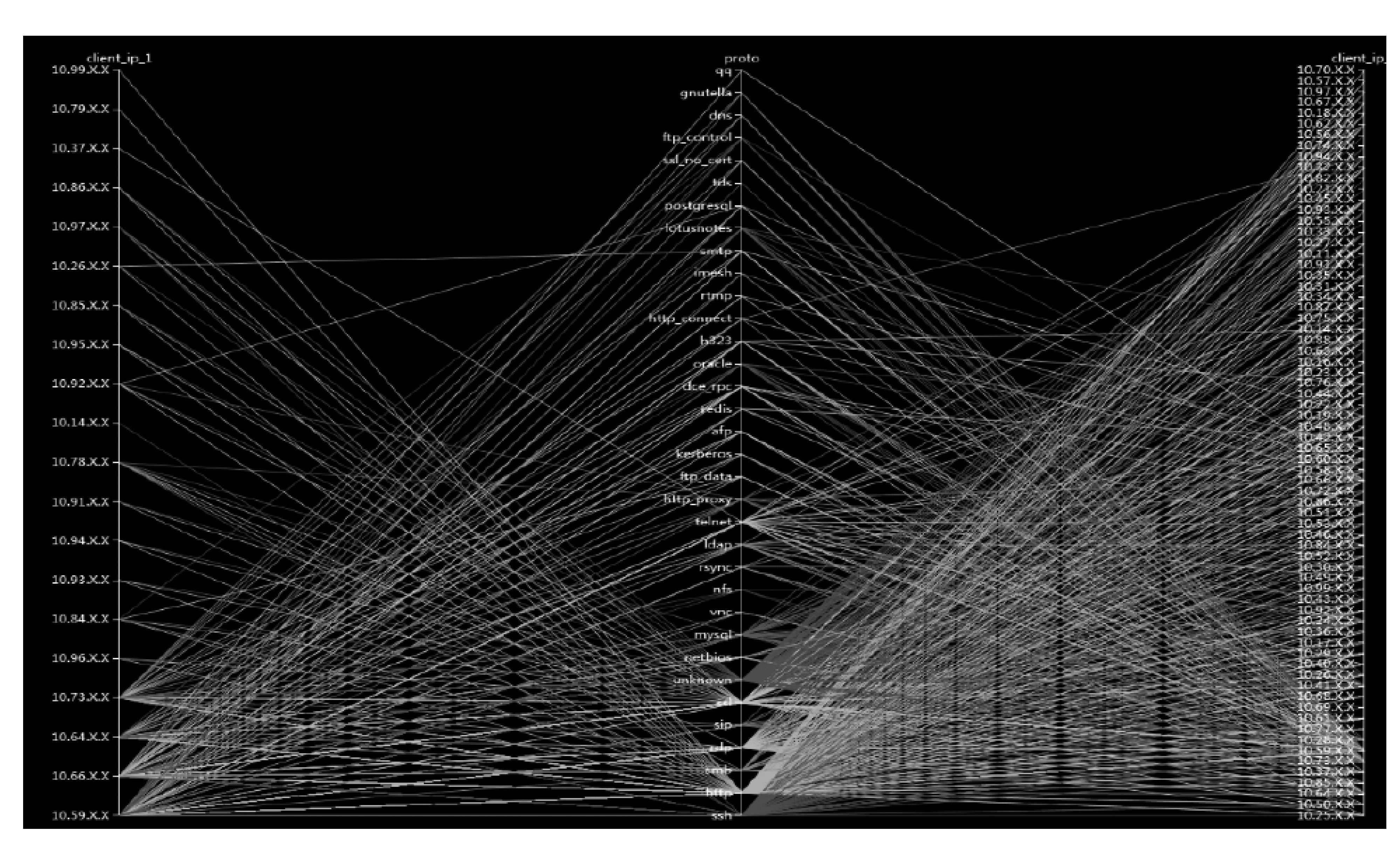


图 3-28　客户端与客户端通信的平行坐标系

5. **模拟实时通信**

在完成预设问题之后，结合所有的数据，对整个内部网络中的通信进行模拟，试图在模拟过程中发现网络中流量的变化规律。使用图 3-29 来模拟内部网络中的实时通信流量。其中选取了一天内的通信数据，之后按照数据包的时间戳进行排序。在程序运行时，

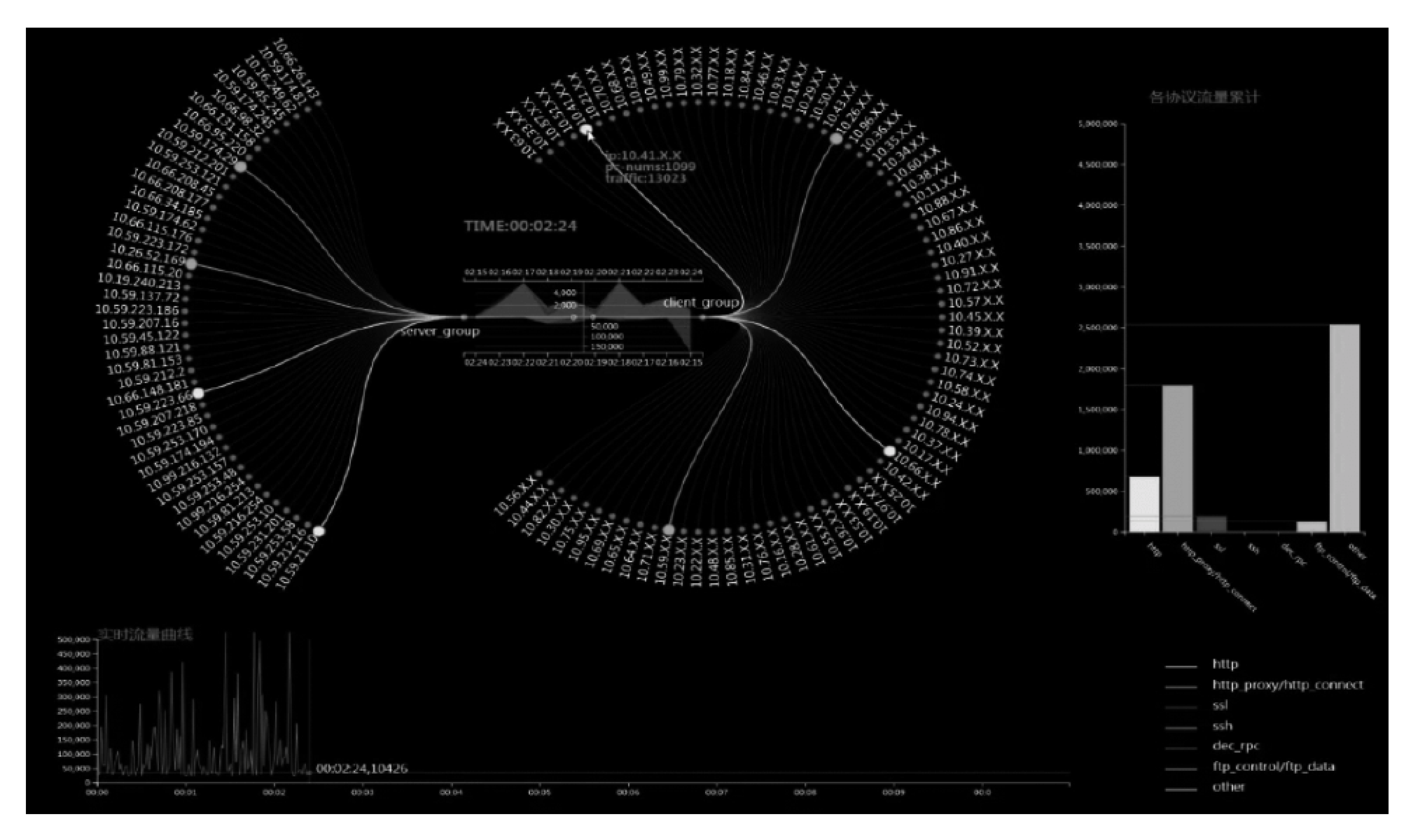


图 3-29　模拟内部网络的实时通信流量

依次读取对应时间的数据，并将其可视化。

在图 3-29 中，使用两个扇形布局分别表示客户端分组和服务器，客户端和服务器之间的每次通信都会经过之间的堆叠图。堆叠图分为上下两个部分，上半部分表示服务器发送给客户端的流量，下半部分表示客户端发送给服务器的流量。并使用不同的填充颜色表示不同的协议。

在图 3-29 下半部分，使用折线图实时记录时间与网络中流量的关系，可以分析出在一段时间之内流量的变化规律。右侧柱状图表示经过不同协议传输数据流量的累积。

3.4.2 案例二：VAST Challenge 2016 竞赛题

1. 简介

VAST Challenge 2016 要求参赛者对 GAStech 公司建筑物内人员流动以及传感器数据进行分析，建筑物总共有 3 层，每一层都被分为不同的区域，包括 prox 和 HVAC 两种区域。要求参赛者根据提供的数据从中发现员工一些典型的行为模式，并对数据进行分析，从中发现一些异常事件。

2. 数据

- 建筑物平面图，包括 prox 区域和 HVAC 区域。
- 雇员列表，包括雇员姓名、职务和办公室标记。
- 数据格式描述。
- prox 卡记录。
- 每个 HVAC 区域内的传感器读数。
- Hazium 传感器读数。

3. 问题

(1) prox 卡的典型模式是什么？一个 GAStech 员工一天的典型行为模式是什么？
(2) 描述 10 种最有趣的模式，并解释这些模式可能的意义。
(3) 描述 10 种值得注意的异常事件，尤其是那些可能进行危险操作的事件。
(4) 描述 5 种 prox 卡和其他数据之间的关系，并分析可能的因果关系。

4. 解答

根据题目要求，需要对雇员和传感器两类数据进行分析，其中雇员数据主要包括雇员的 prox 行为轨迹以及部门分布。传感器数据包括 3 个楼层的设备传感器、通用传感器以及 Hazium 传感器。将主界面分为左右两大块区域，左侧对雇员数据可视化，右侧对各类传感器进行可视化，如图 3-30 所示。

在左侧，对雇员数据可视化时，使用散点图表示雇员的活动规律。横轴表示时间，纵轴表示出入区域。并且将不经过楼梯/电梯而直接穿越不同楼层的活动定义为异常轨迹，异常轨迹使用黄色圆点标注在时间轴上。

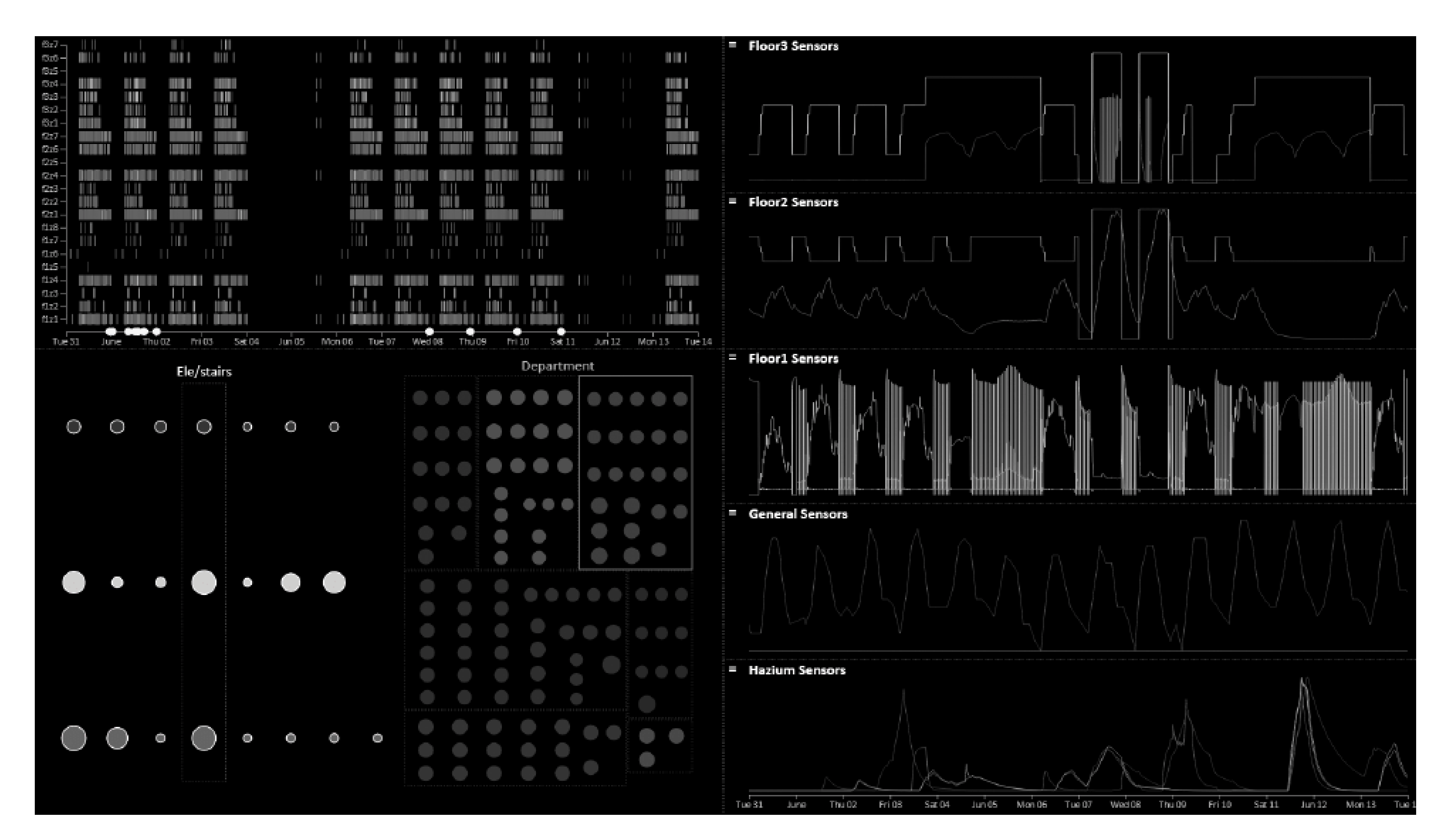


图 3-30　整体设计图

在右侧对传感器数据可视化时，考虑到传感器数量过多，不宜同时展示，因而设计了交互。通过交互选择指定的传感器并进行展示，进而分析传感器读数与时间的关系来解答预设的问题。

（1）prox 卡的典型模式是什么？一个 GAStech 员工一天的典型行为模式是什么？

在图 3-31 中，将 prox 卡数据以散点图的形式可视化，其中竖轴表示建筑物内所有的 prox 区域，使用横轴表示时间。将人员的活动编码为不同颜色的散点。在图 3-31 中，可以发现以下几种基本的模式：

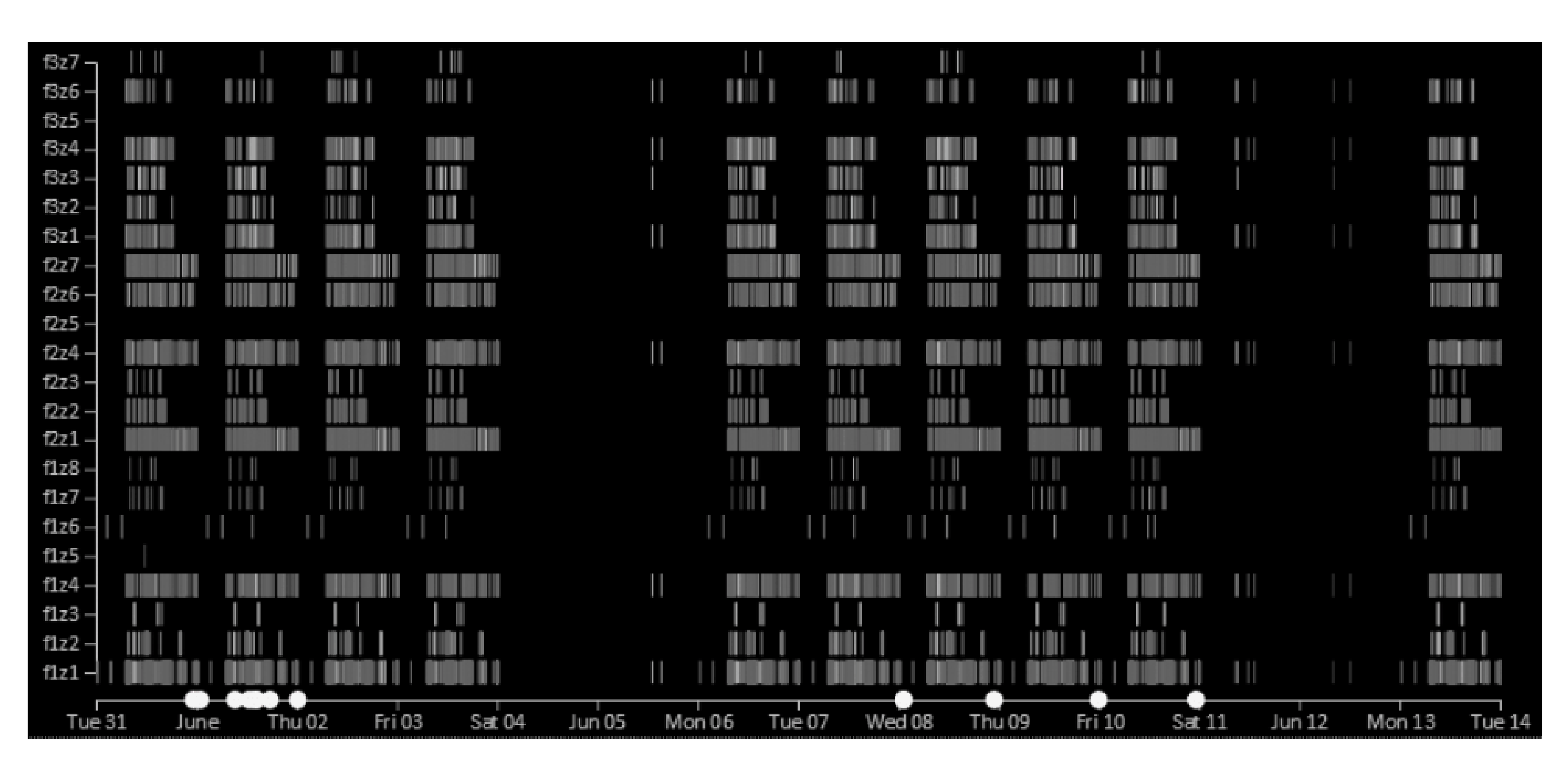


图 3-31　prox 卡数据的散点图

• 公司内的工作日为每周的周一到周五。
• 工作时间从上午 7 点到晚上 24 点。
• 在周末有雇员轮流值班。

通过分析雇员的轨迹，可以发现大多数雇员只在固定的某些区域活动。例如，在图 3-32 中，可以看出雇员从未在三层活动，而是大多数时间出现在二层。而图 3-33 中，雇员则经常出现在三楼。因为建筑物的出入口在一楼，所以每个雇员的轨迹都会在第一层出现。

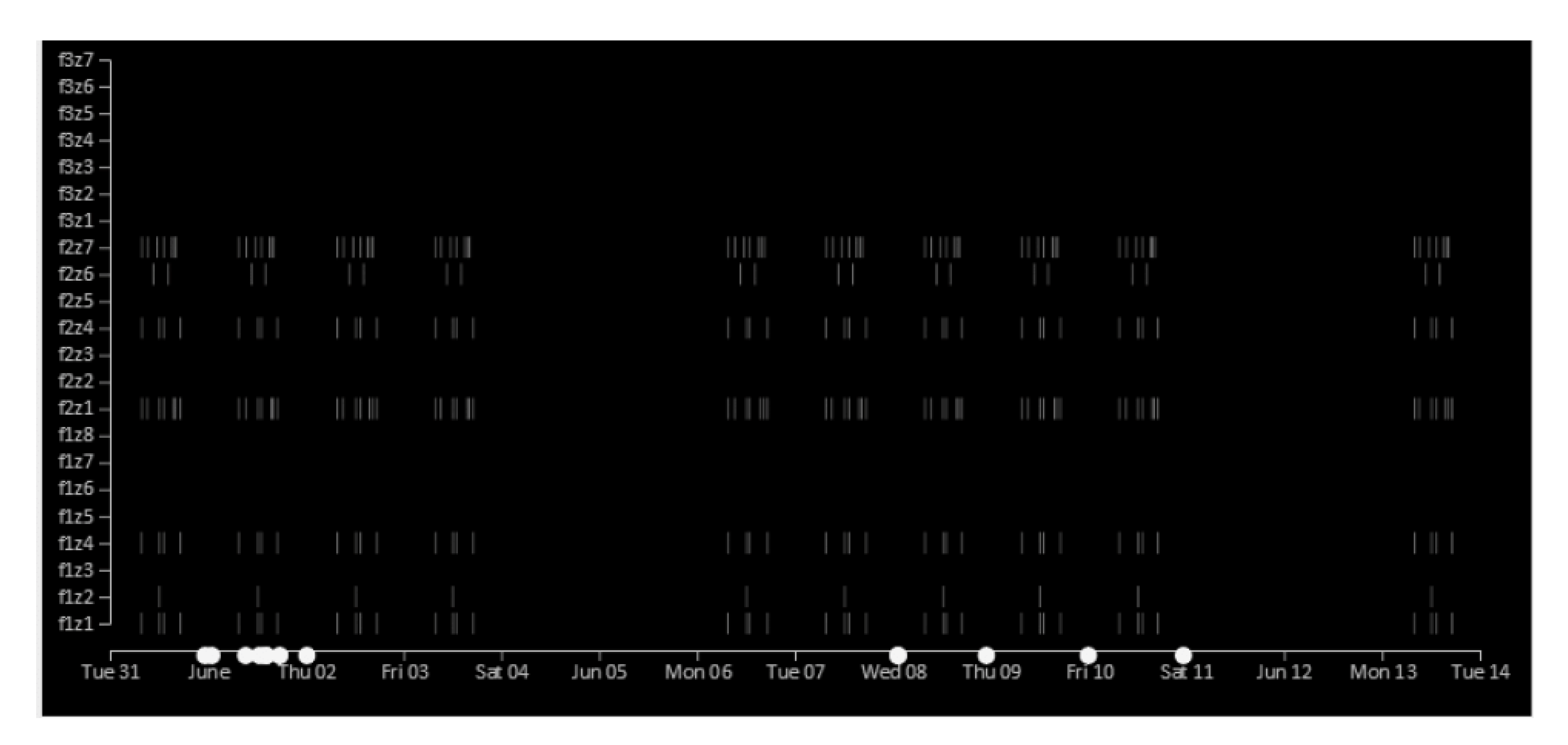


图 3-32 分析雇员轨迹(一)

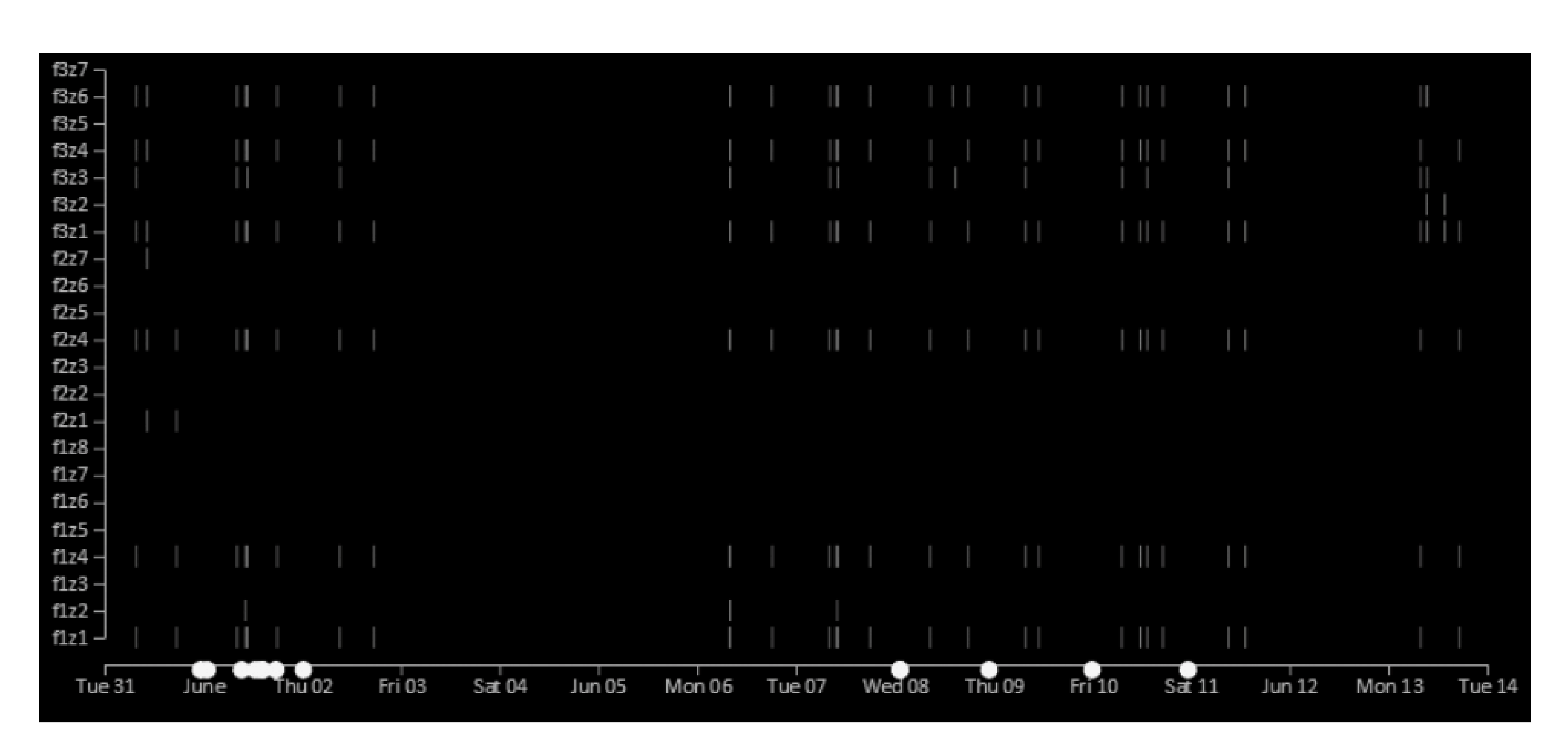


图 3-33 分析雇员轨迹(二)

图 3-34 展示了每个雇员在不同 prox 区域的轨迹。使用不同的颜色表示不同楼层的 prox 区域。红色表示一楼，绿色表示二楼，蓝色表示三楼。点的大小表示经过这个区域的人员流量。使用线条表示雇员的轨迹，同楼层之间的轨迹使用灰色线条表示，不同楼层之间的轨迹使用对应楼层的颜色表示。对雇员轨迹的模拟中可以发现：

• 每个雇员在进入建筑物都会经过一楼的第一个 prox 区域。

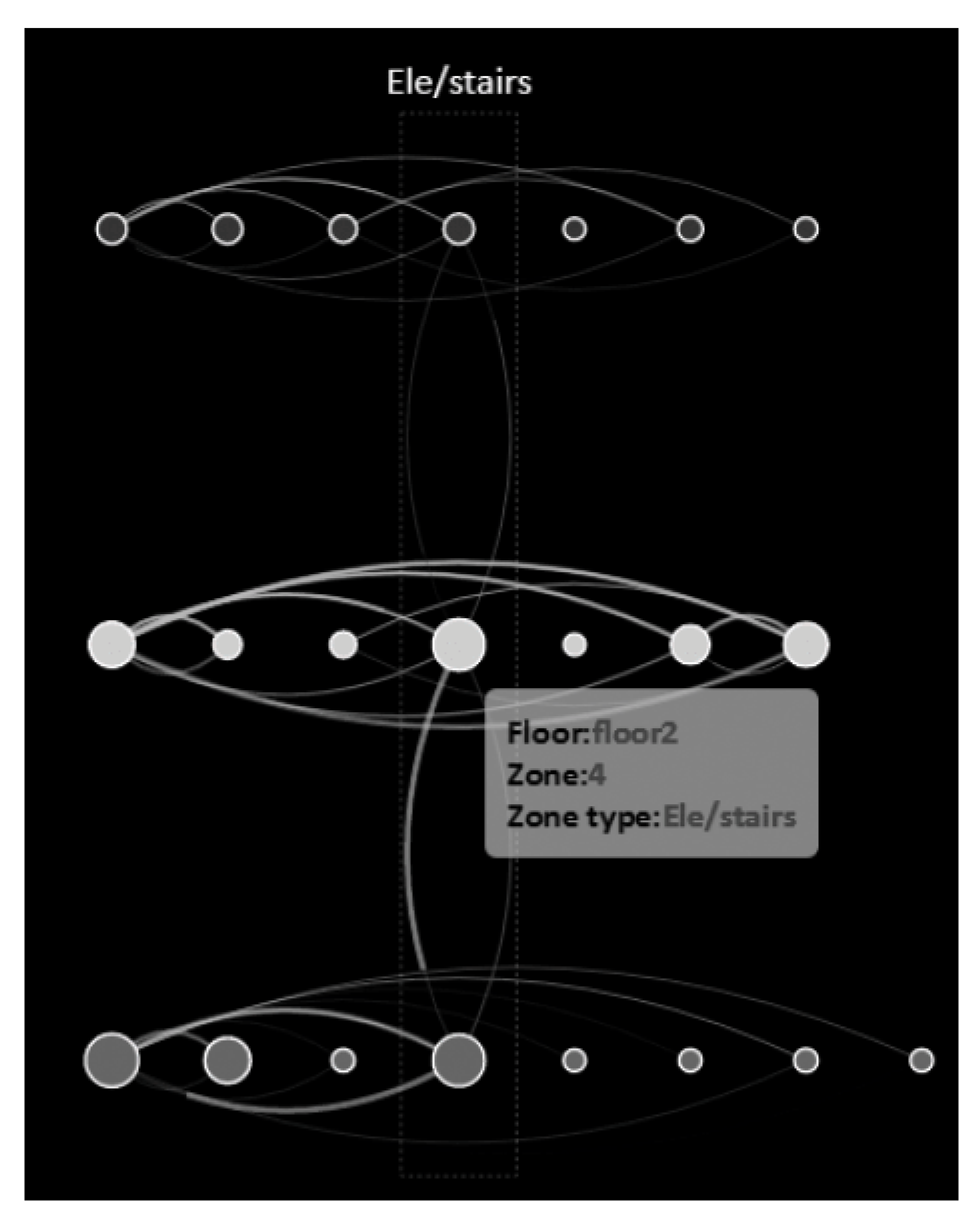


图 3-34 雇员在不同 prox 区域的轨迹图

- 雇员大多数时间都在同一楼层内活动，因为同一楼层内的轨迹明显多于不同楼层之间的轨迹。

（2）描述 10 种最有趣的模式，并解释这些模式可能的意义。

图 3-35 使用折线图展示了 Hazium 传感器数据的读数，可以从图中发现 4 个 Hazium 传感器的变化规律几乎一致，说明这 4 个区域是互通的，其中一个区域的 Hazium 浓度变化会引起其他 3 个浓度的变化。

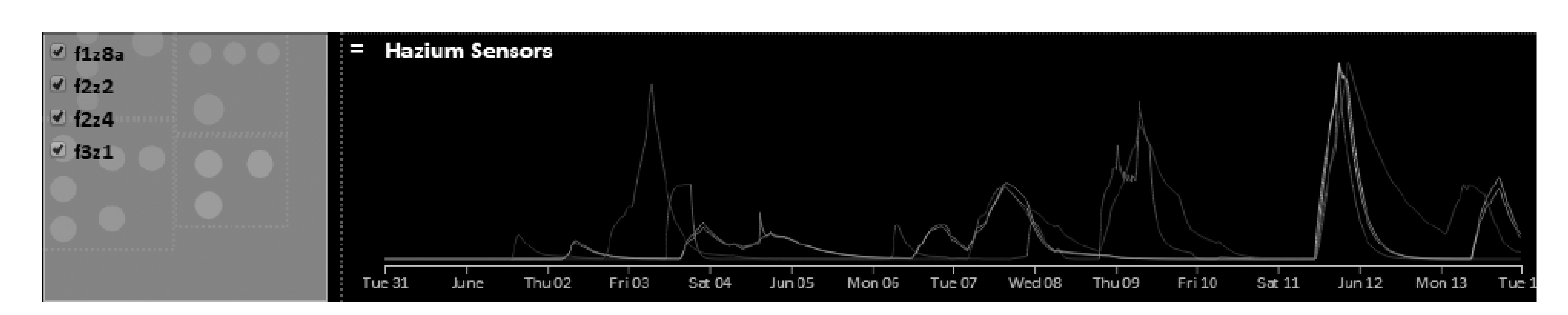


图 3-35 Hazium 传感器数据读数的折线图

图 3-36 展示了热水器燃气消耗和热水器水箱温度的变换规律，从图中可以发现，在上班时间两个传感器的数据会上下波动，而夜晚或者周末则处于稳定状态。

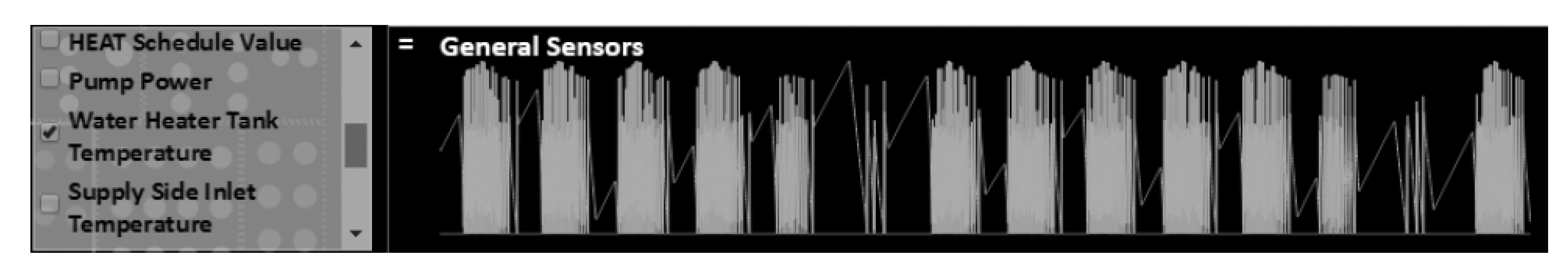


图 3-36 热水器燃气消耗和热水器水箱温度变换规律的折线图

图 3-37 展示了建筑物内所有区域的照明电路的开关，从图中可以看出，这些照明电源也是随着工作时间变化的。

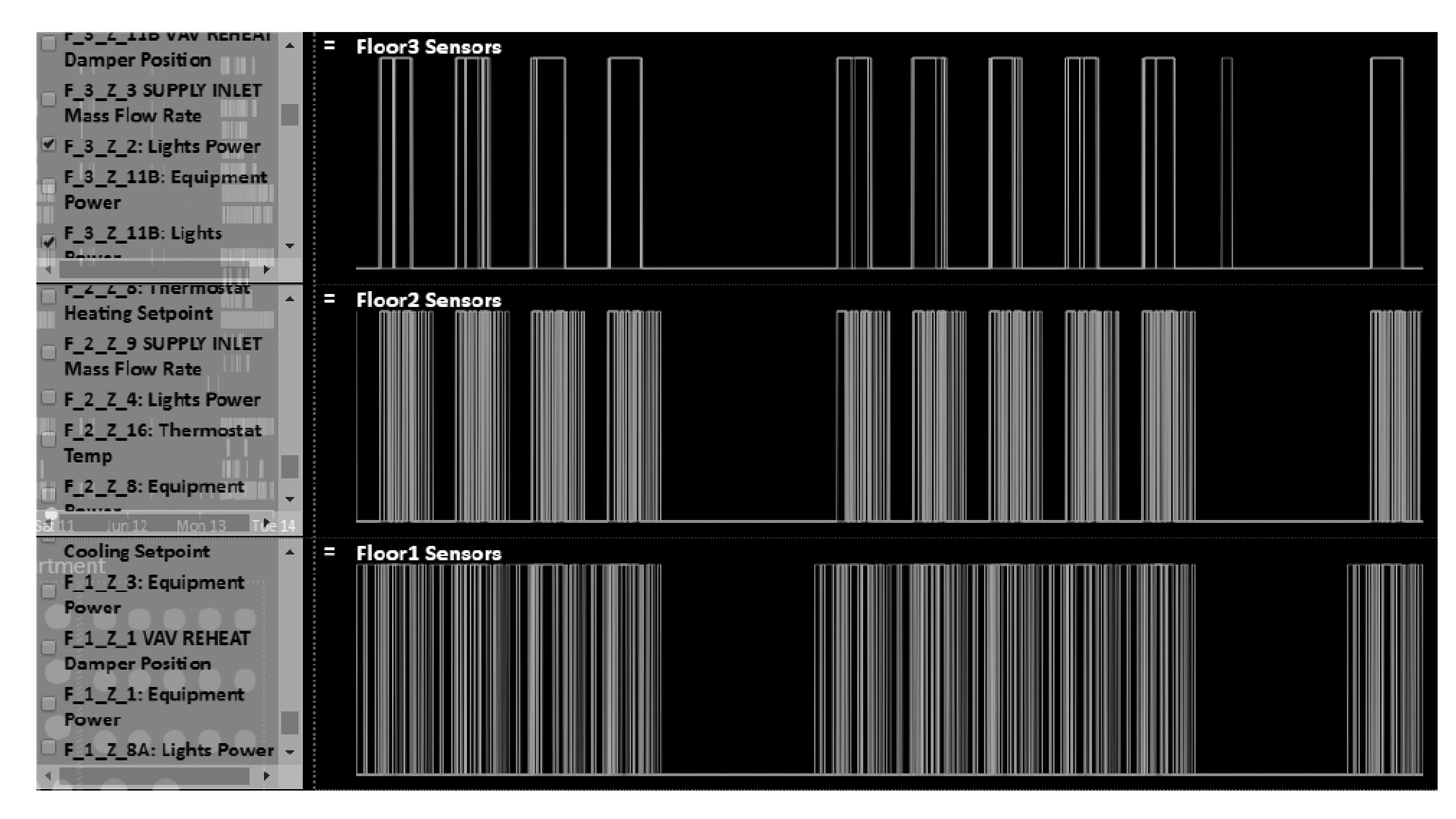


图 3-37　建筑物内所有区域的照明电路开关图

图 3-38 展示了通风口处二氧化碳浓度传感器读数的变化，它可以大致反映员工的数量变化，在工作时间，随着员工数量的增加，二氧化碳的浓度也会随之升高；而在非工作时间，二氧化碳的浓度则会降低。

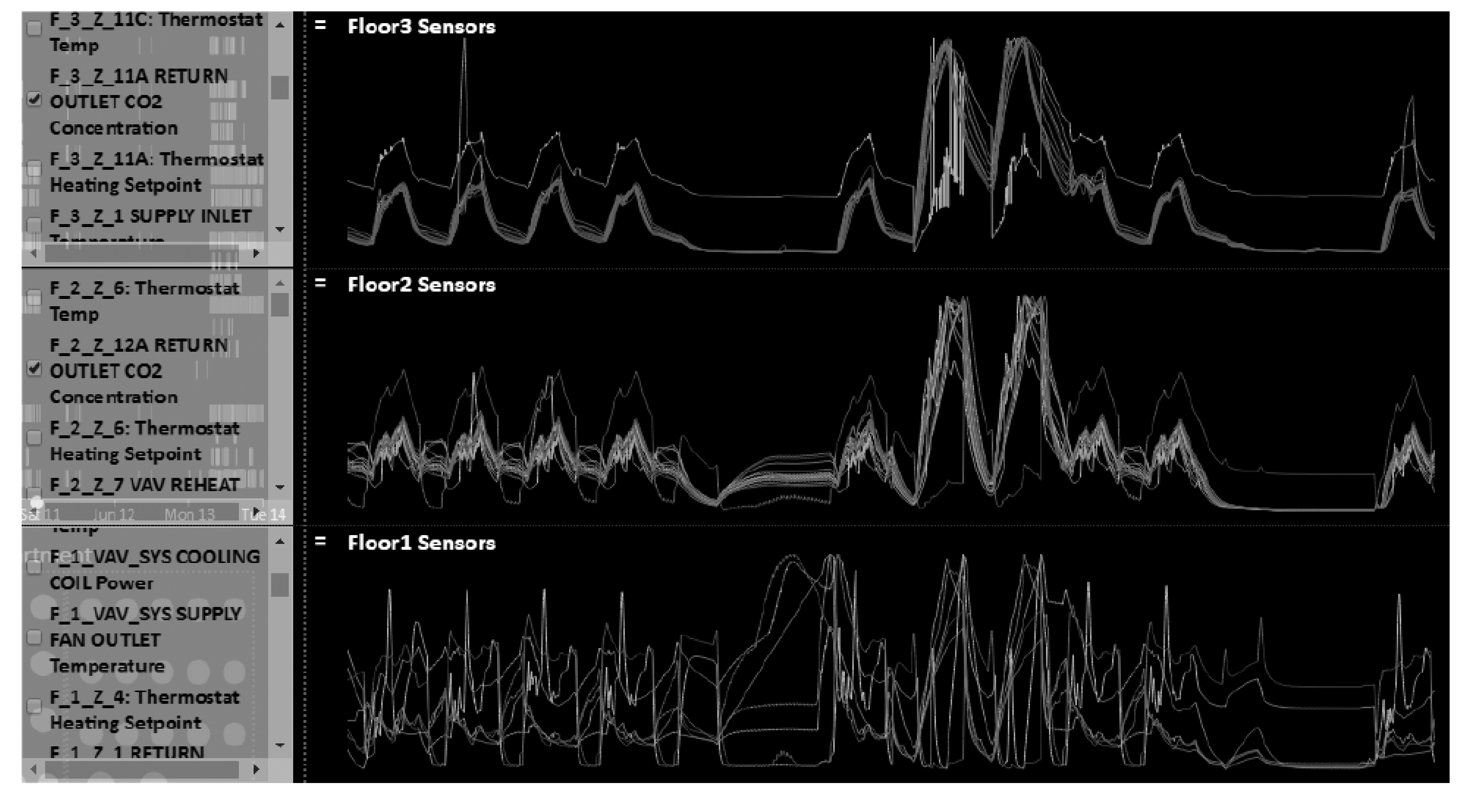


图 3-38　通风口处二氧化碳浓度传感器读数变化的折线图

图 3-39 展示了建筑物内设备用电量的变化规律，它反映了电力设备在工作时间内的周期性，即工作时间内的功耗比下班时间大得多。

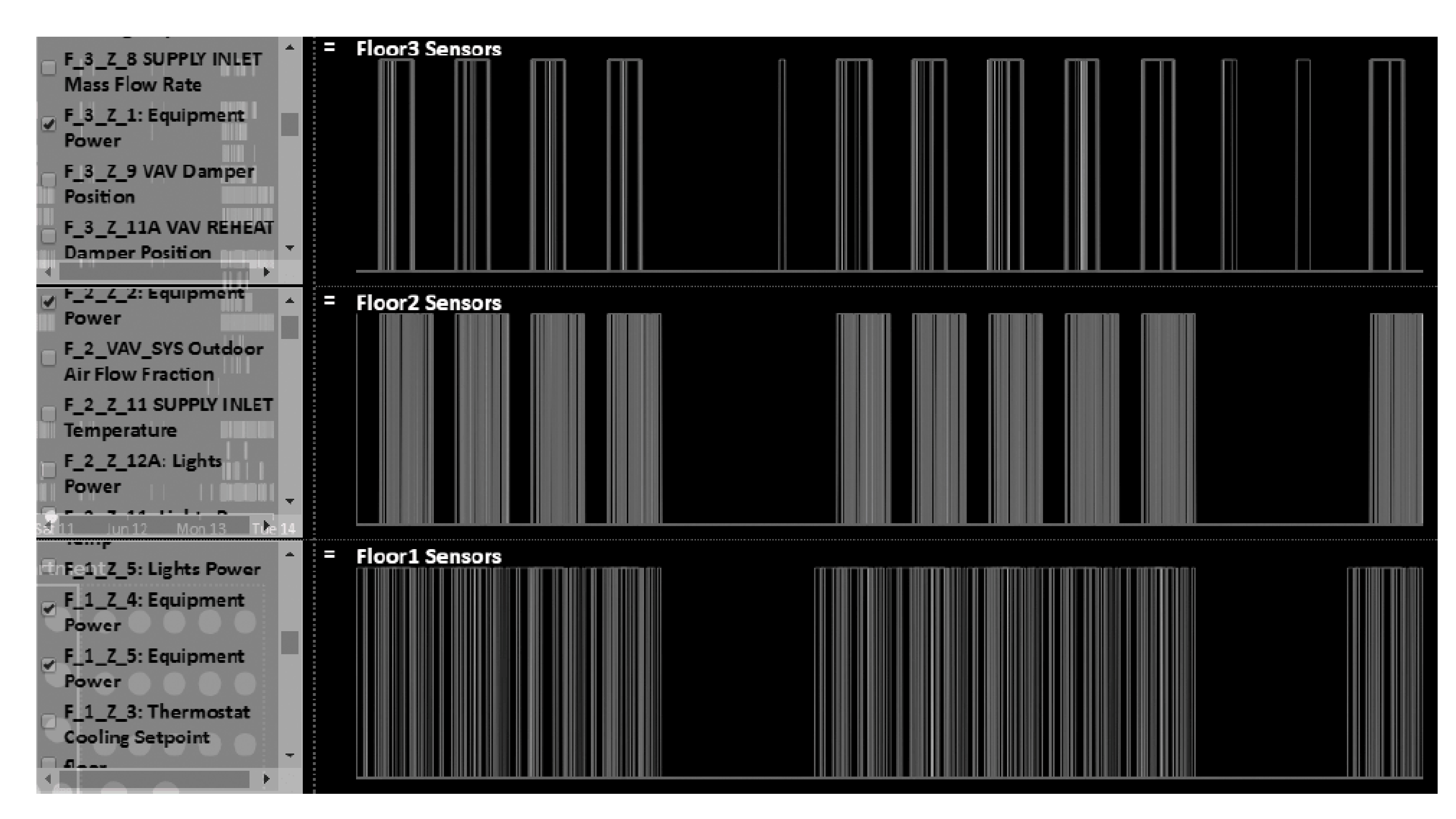


图 3-39 建筑物设备用电量变化规律图

图 3-40 反映了干球温度传感器的读数变化，从图中可以看出这个温度传感器的读数变化比较稳定，说明在这段时间内没有由温度变化引起的异常。

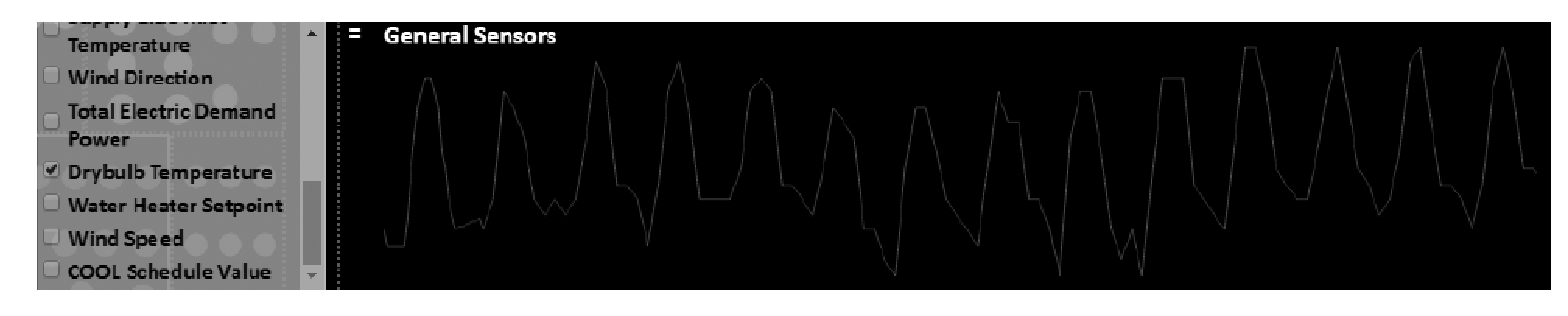


图 3-40 干球温度传感器的读数变化图

图 3-41 展示了不同部门的雇员分布，从图中可以看出工程部的人数多于其他部门的人数，而人力资源部仅有 3 个雇员。

图 3-42 中对一层的区域 6 分析发现，在每个工作日，都会有同一个雇员出现在这个区域，查询建筑平面图发现这个区域为配置室。可以看出，建筑物内的设施都有专门的维护人员进行维护。

图 3-43 展示了 HVAC 区域的用电量和 Hazium 之间的关系，从中可以发现，当 Hazium 浓度升高时，HVAC 区域的用电量也会升高，可以看出，在 Hazium 浓度升高时建筑物内会启动某些设备进行通风。

图 3-44 展示了 prox 卡的散点图，从图中可以发现在两周内没有人员出现在三层的区域 5 和二层的区域 5。在查询设计图之后发现，三层的区域 5 是一个未开发区域，因此并没有人员进出；而二层的区域 5 推测可能是一个特别区域，一般的员工并没有权限进入这个区域。

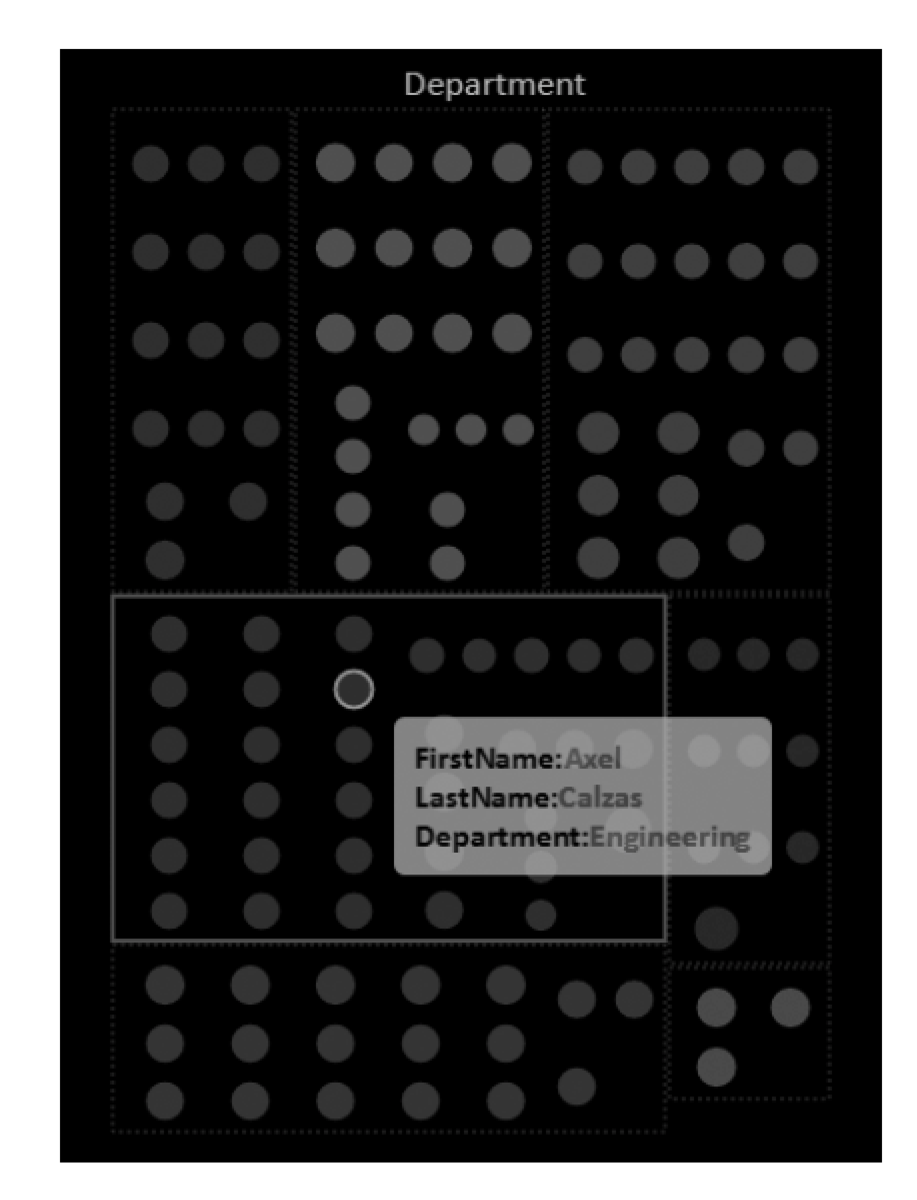


图 3-41 不同部门的雇员分布

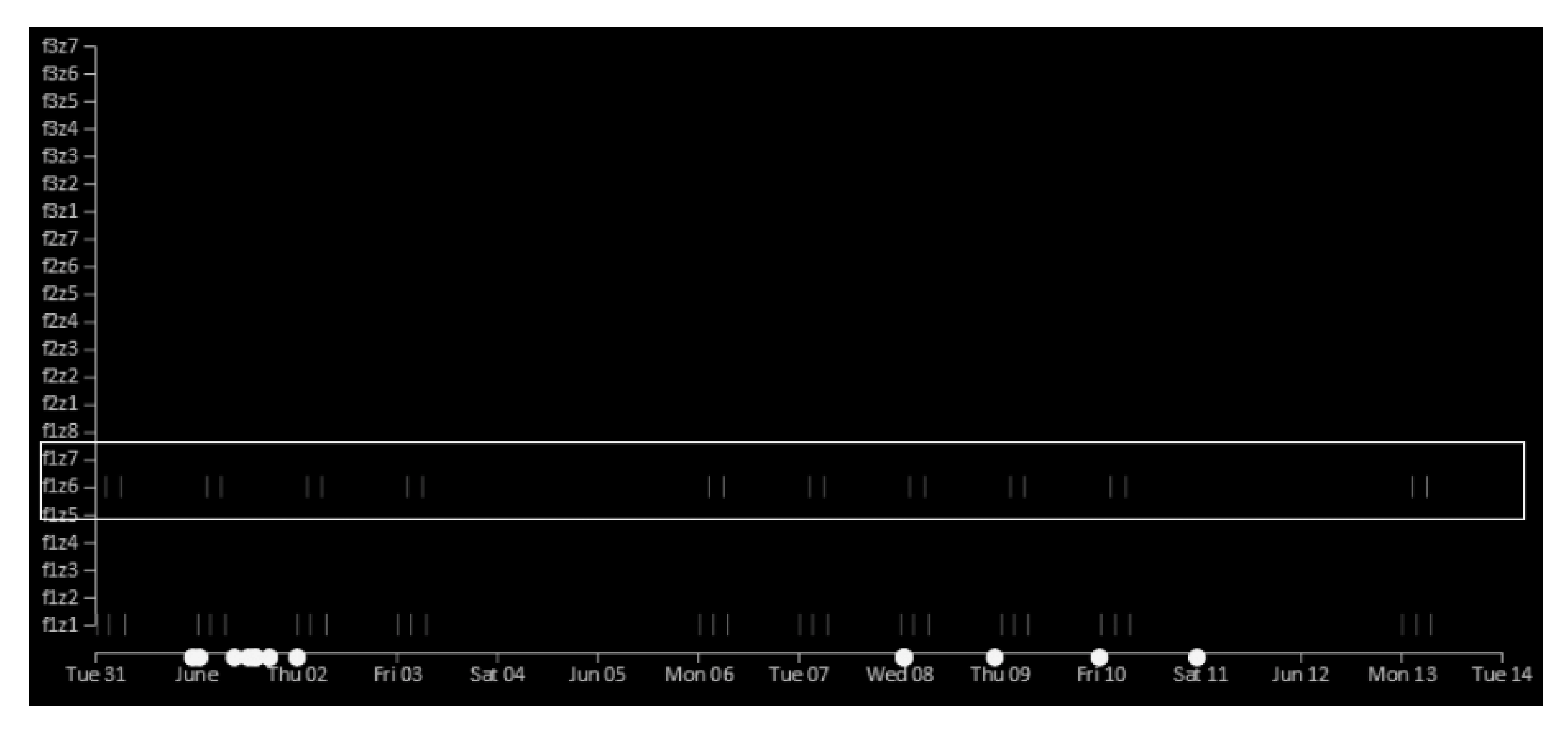


图 3-42 对一层区域 6 的分析图

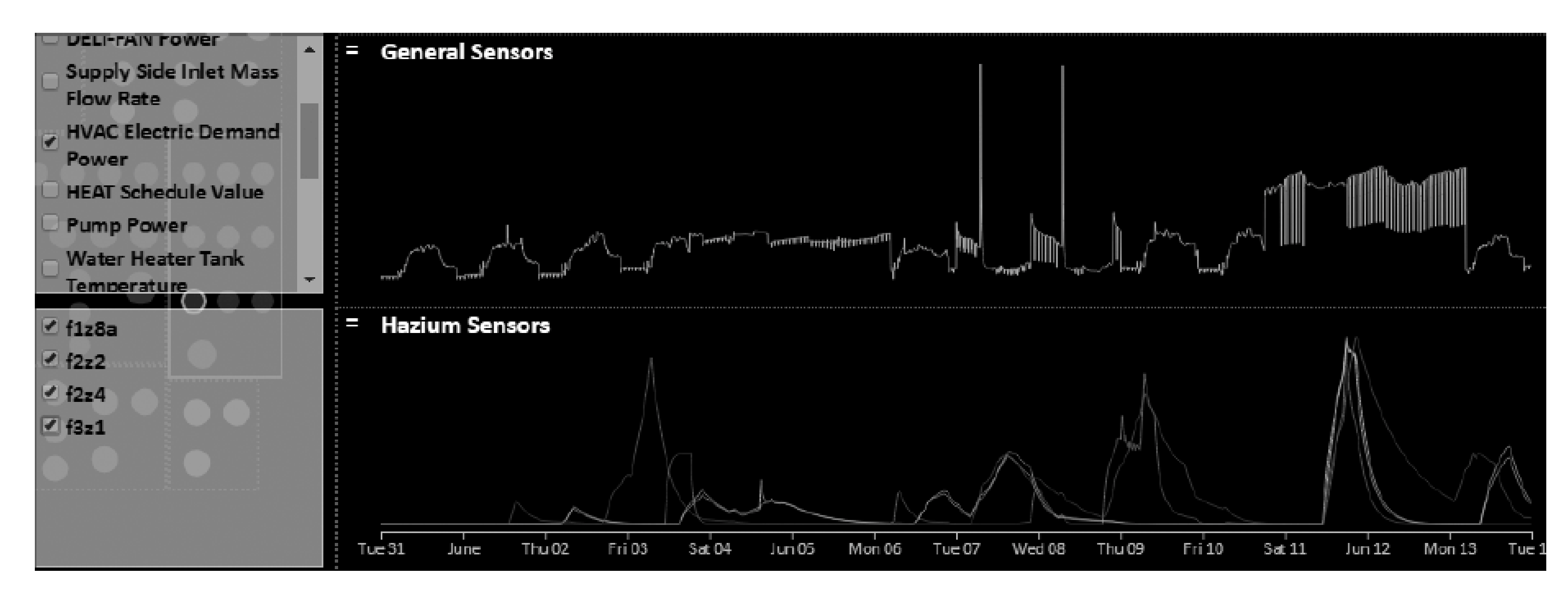


图 3-43 HVAC 区域的用电量和 Hazium 浓度之间的关系图

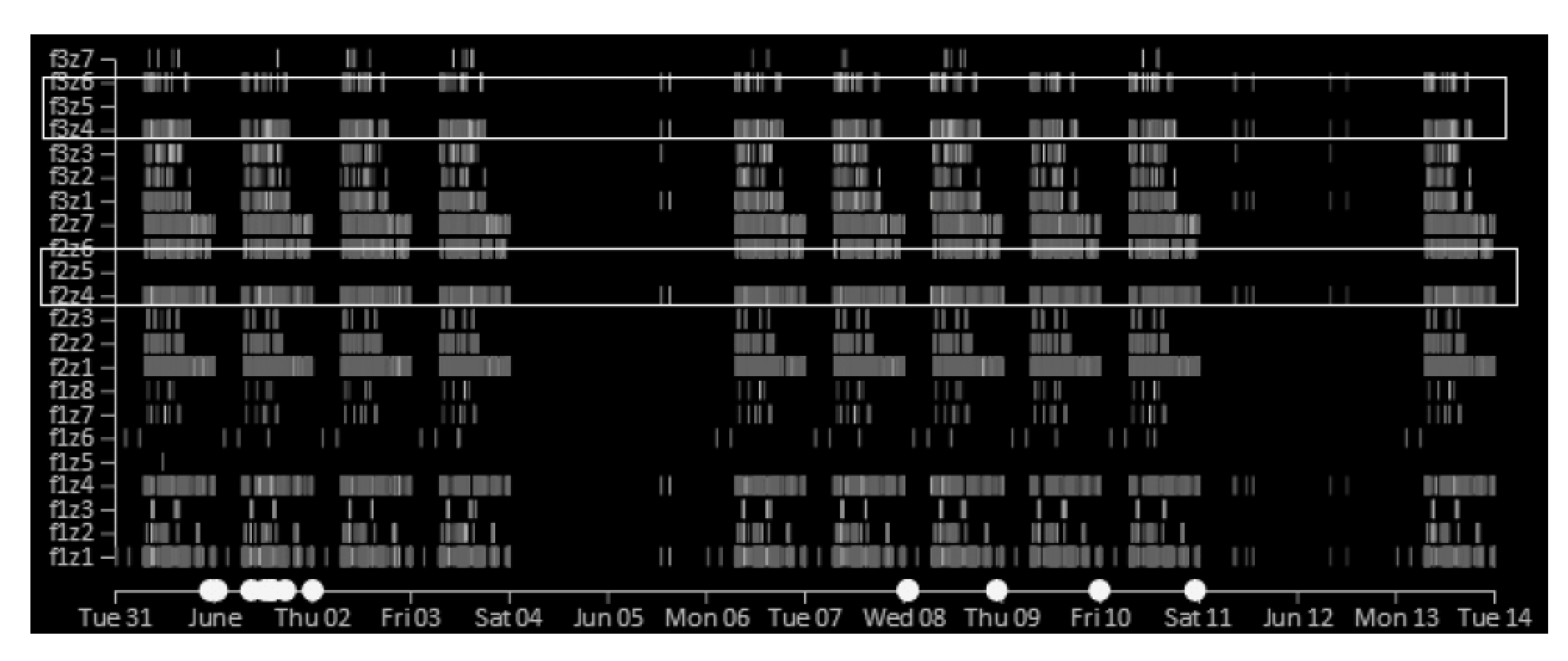


图 3-44　prox 卡散点图

（3）描述 10 种值得注意的异常事件，尤其是那些可能进行危险操作的事件。

从图 3-45 中可以发现，在 6 月 1 日 1 点左右，ID 为 ibaza001 和 edavies001 的 prox 卡在没有经过楼梯的情况下直接从一楼的区域 1 到二楼的区域 1，这可能为一个异常的时间。

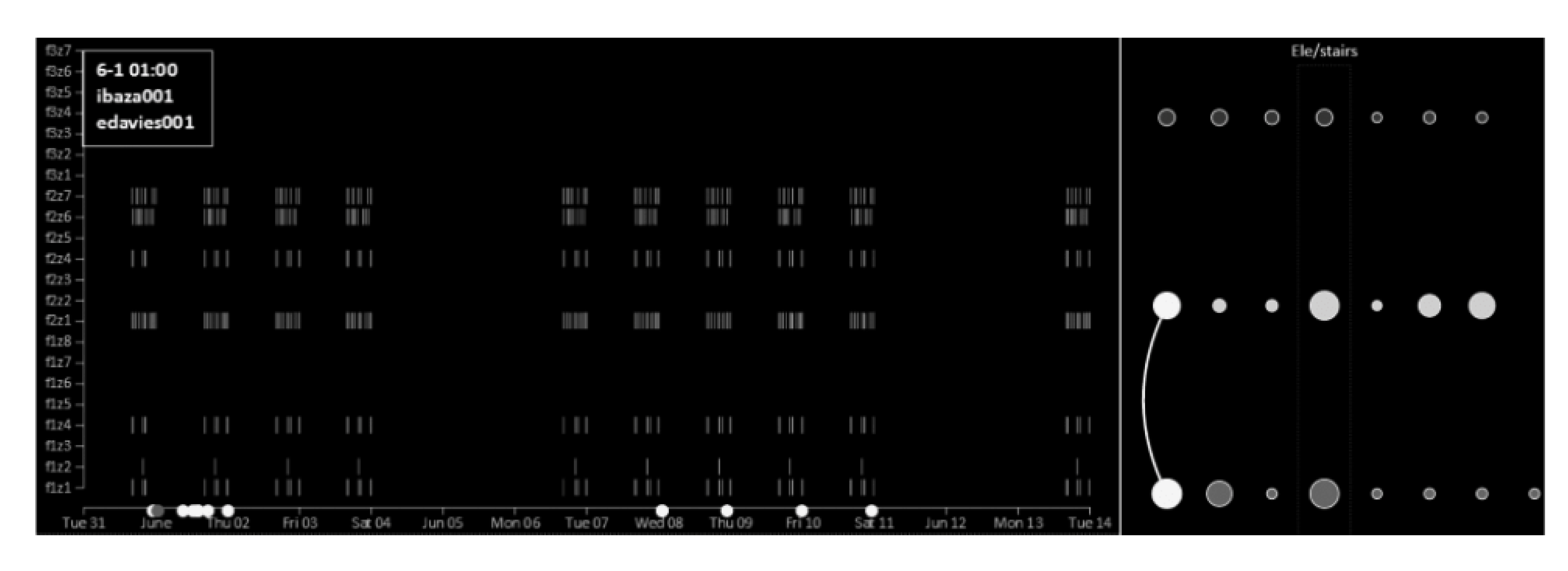


图 3-45　对 ibaza001 和 edavies001 活动异常的分析

从图 3-46 可以看出，6 月 7 日 12 点到 8 日 11 点，SUPPLY INLET Temperature 传感器读数明显高于其他时间段，因此这个时间段发生了异常，并且这个异常可能是由于 HVAC 系统的异常导致的。



图 3-46　SUPPLY INLET Temperature 传感器读数异常

从图 3-47 可以看出，一层的区域 2 的设备功率传感器明显与其他时间段不同，也将其定义为一个异常事件，并且推测可能是与建筑物内其他异常事件有关联。



图 3-47　一层的区域 2 设备功率传感器异常

图 3-48 中，电源功率在 6 月 7 日的 7 点和 8 日的 7 点的读数高于其他时间段，因此推测在这两个时间点出现了异常事件导致某些设备自动启动。

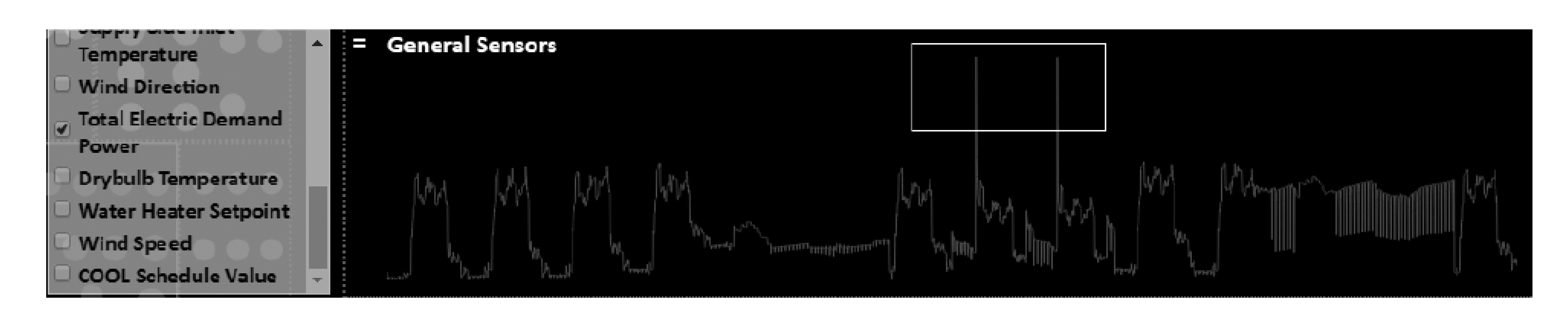


图 3-48　电源功率异常

图 3-49 展示了 Availability Manager Night Cycle Control Status 的读数，对比 3 个楼层的读数可以发现，一层和二层的读数都符合工作时间规律，而三层的读数则一直没有变化，据此推测三层的这个设备可能损坏或被人为关闭。

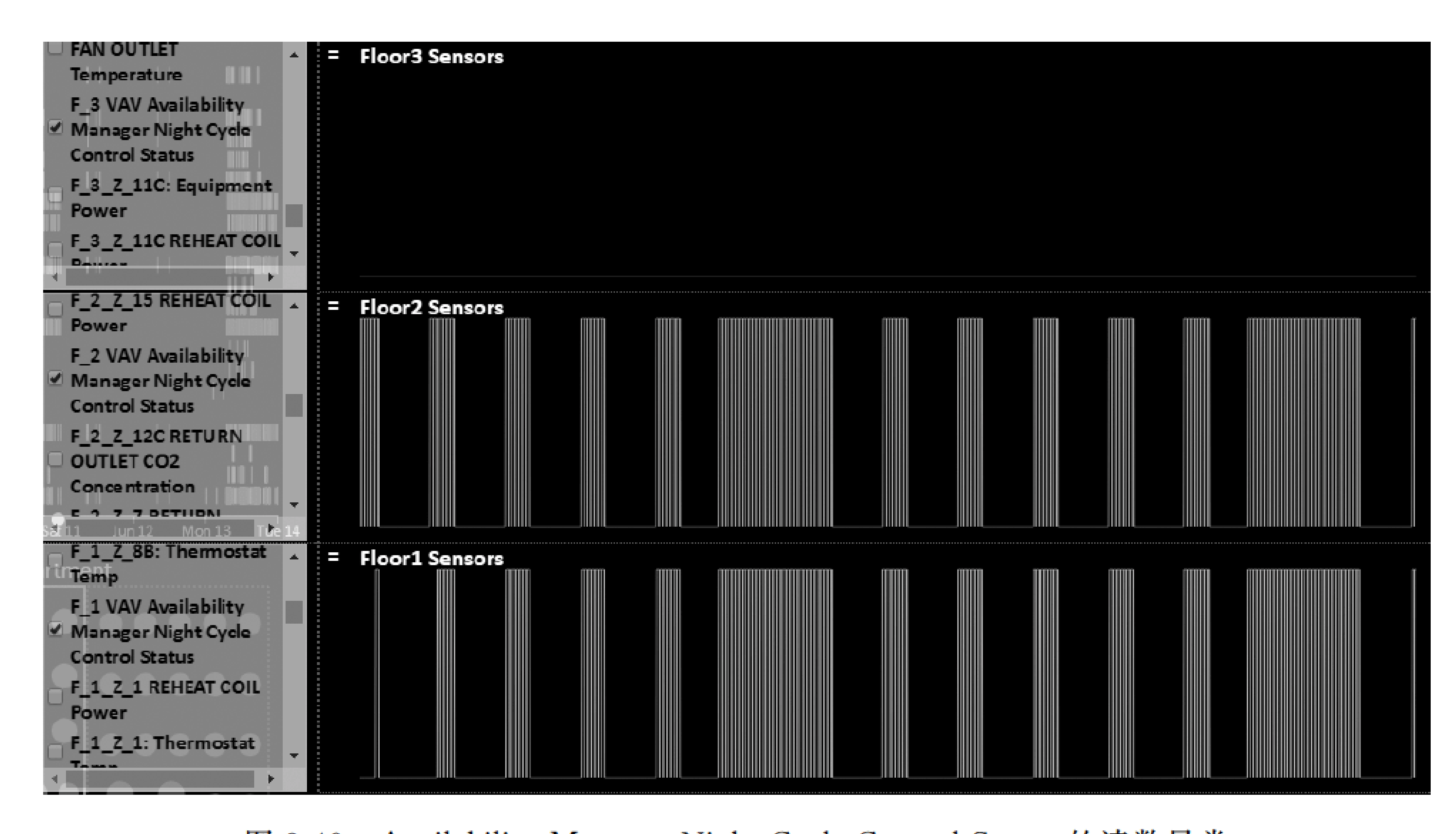


图 3-49　Availability Manager Night Cycle Control Status 的读数异常

图 3-50 展示了冷却系统的功率，对比可以发现 3 个楼层的这些传感器的读数在 6 月 7 日和 8 日都出现了升高。因此可以推测这两天发生了某些异常事件，导致建筑物内冷却系统用电量升高。

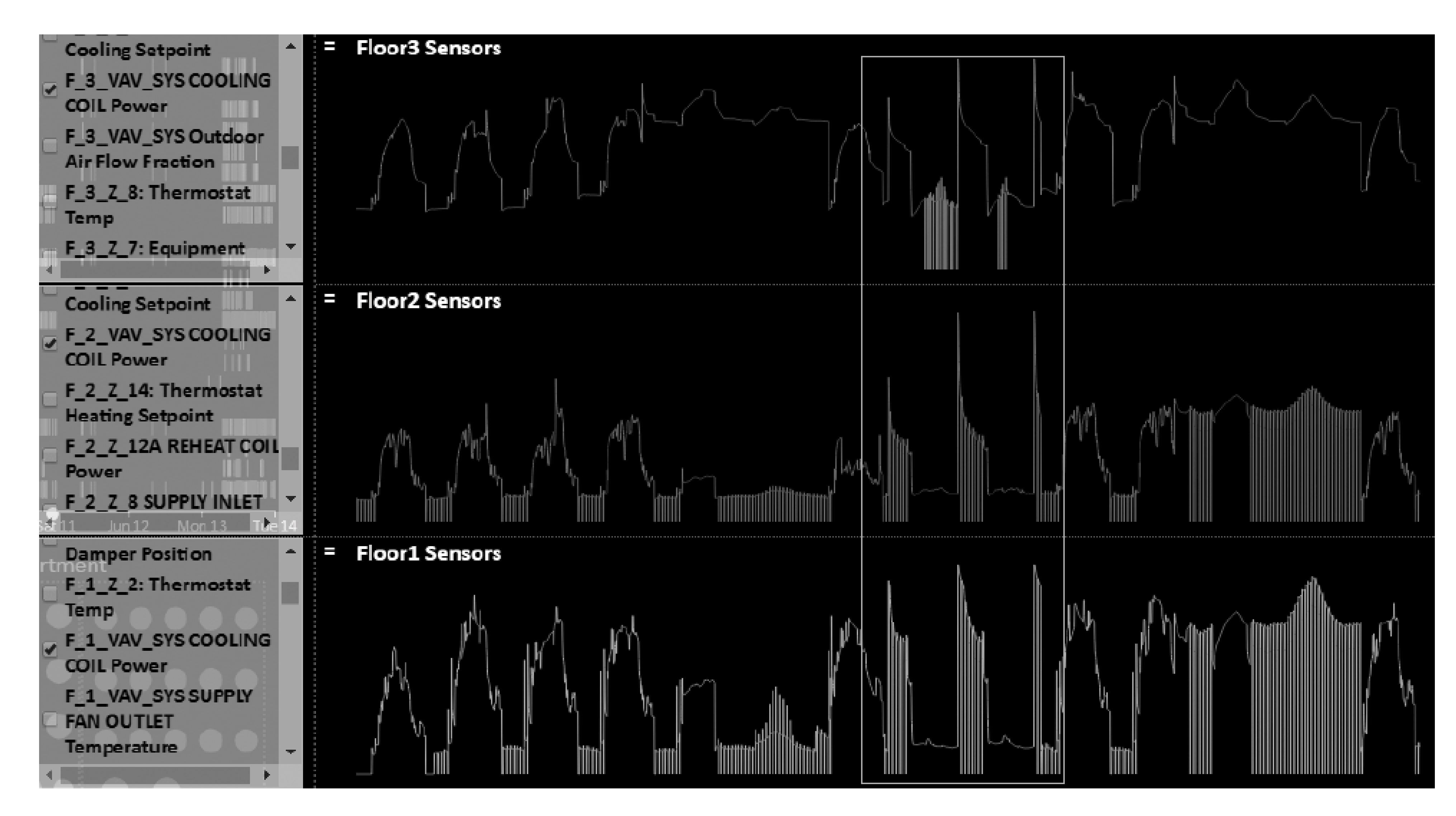


图 3-50　冷却系统的功率异常

图 3-51 展示了 3 个楼层的风扇用电功率，在周末的时候，二层和三层的功率下降，而三层的功率却上升，说明在周末的时候三层出现了异常事件导致风扇用电量增加。

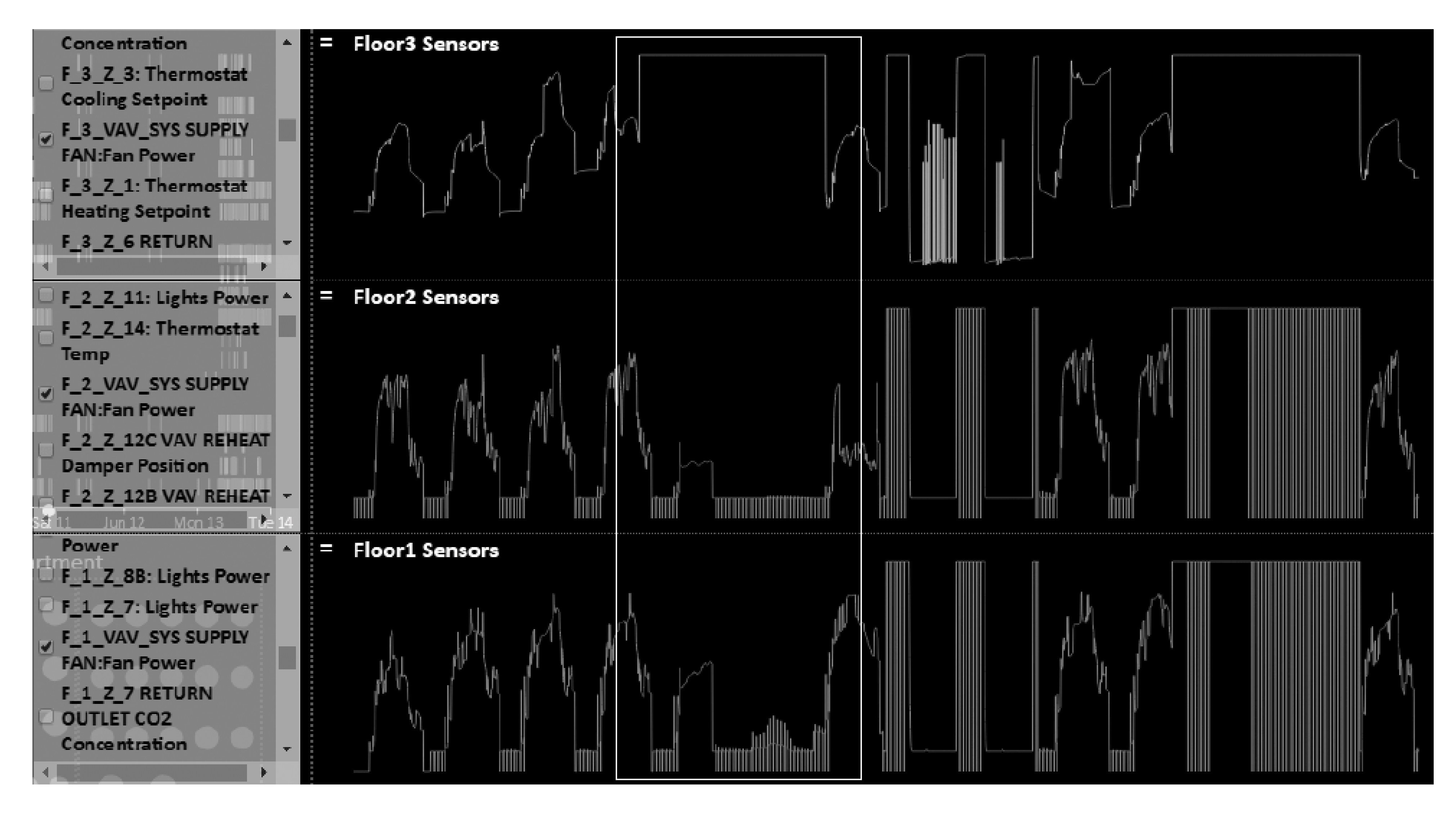


图 3-51　3 个楼层的风扇用电功率异常

在图 3-52 所示的人员活动散点图中，可以发现在第二个周末时，建筑物内除了值班人员之外还有其他人员活动，结合 Hazium 浓度可以发现，在同一时间段，Hazium 的浓度也上升了，因此推测可能是这些人员的行为导致 Hazium 浓度上升。

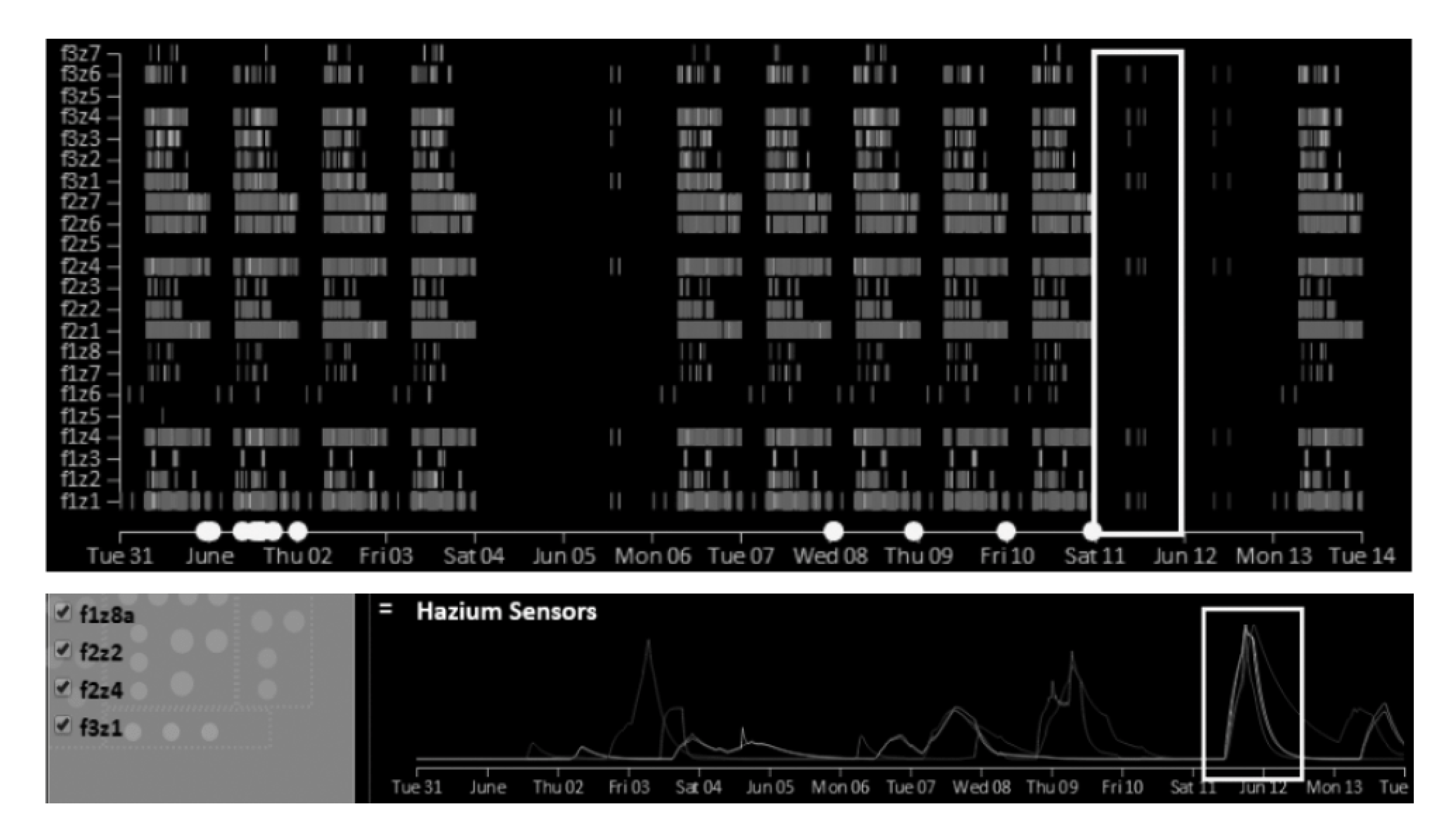


图 3-52　人员活动散点图

图 3-53 展示了温度控制器冷却值读数变化，在 6 月 7 日到 9 日，3 个楼层的读数变化明显与其他时间段的变化规律不同，因此推测在这个时间段出现了异常事件导致温度控制器的读数发生变化。

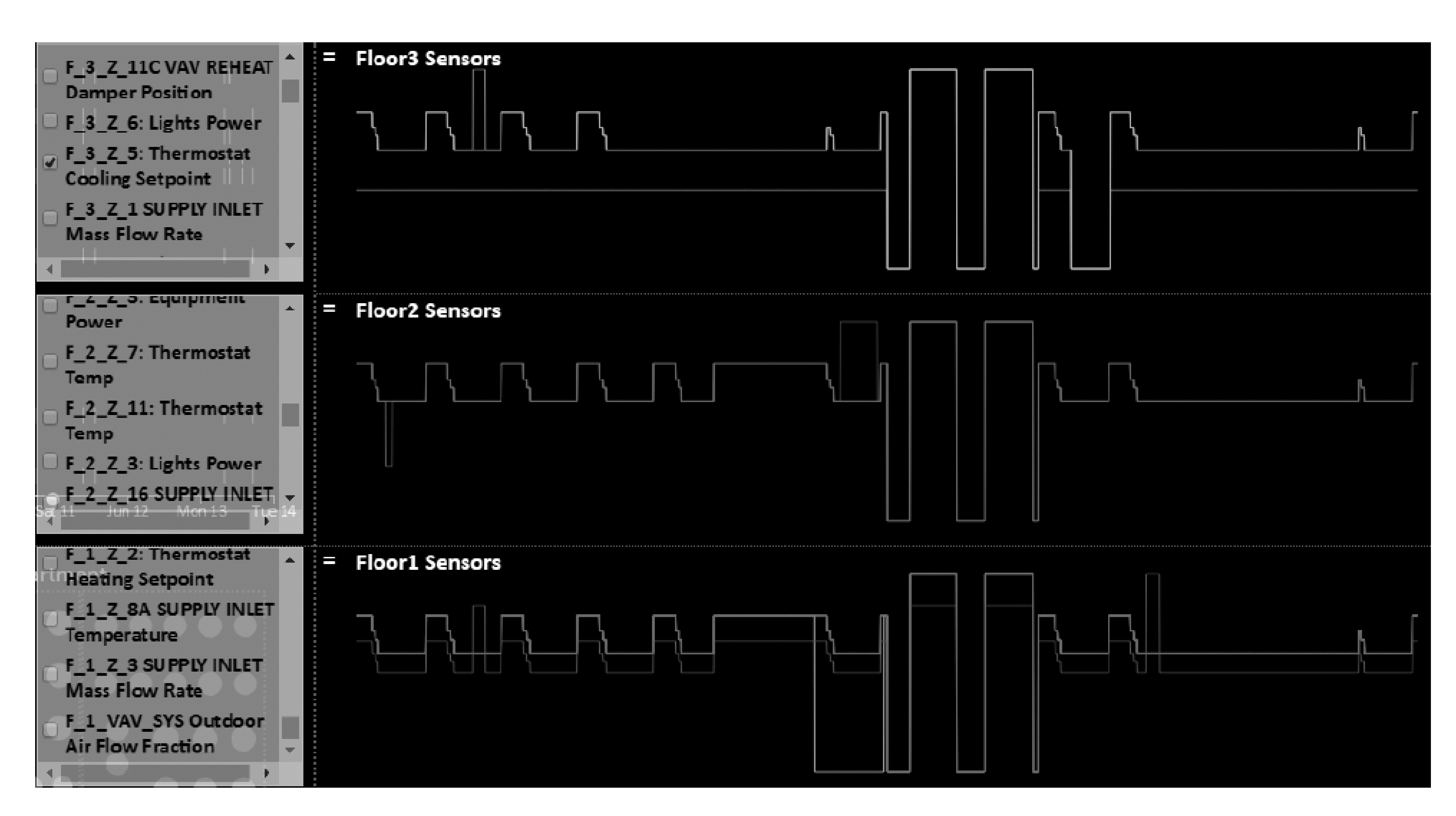


图 3-53　温度控制器冷却值读数异常

在图 3-54 中，对比 3 个楼层的系统冷却电源用电量和 Hazium 浓度变化发现，从 6 月 11 日 7 点开始，在 Hazium 的浓度开始升高的同时，3 个楼层的冷却系统用电量也随之升高，可以推测，当 Hazium 浓度升高时，系统会自动启动某些设备来调节浓度的变化，因此用电量也会随之升高。

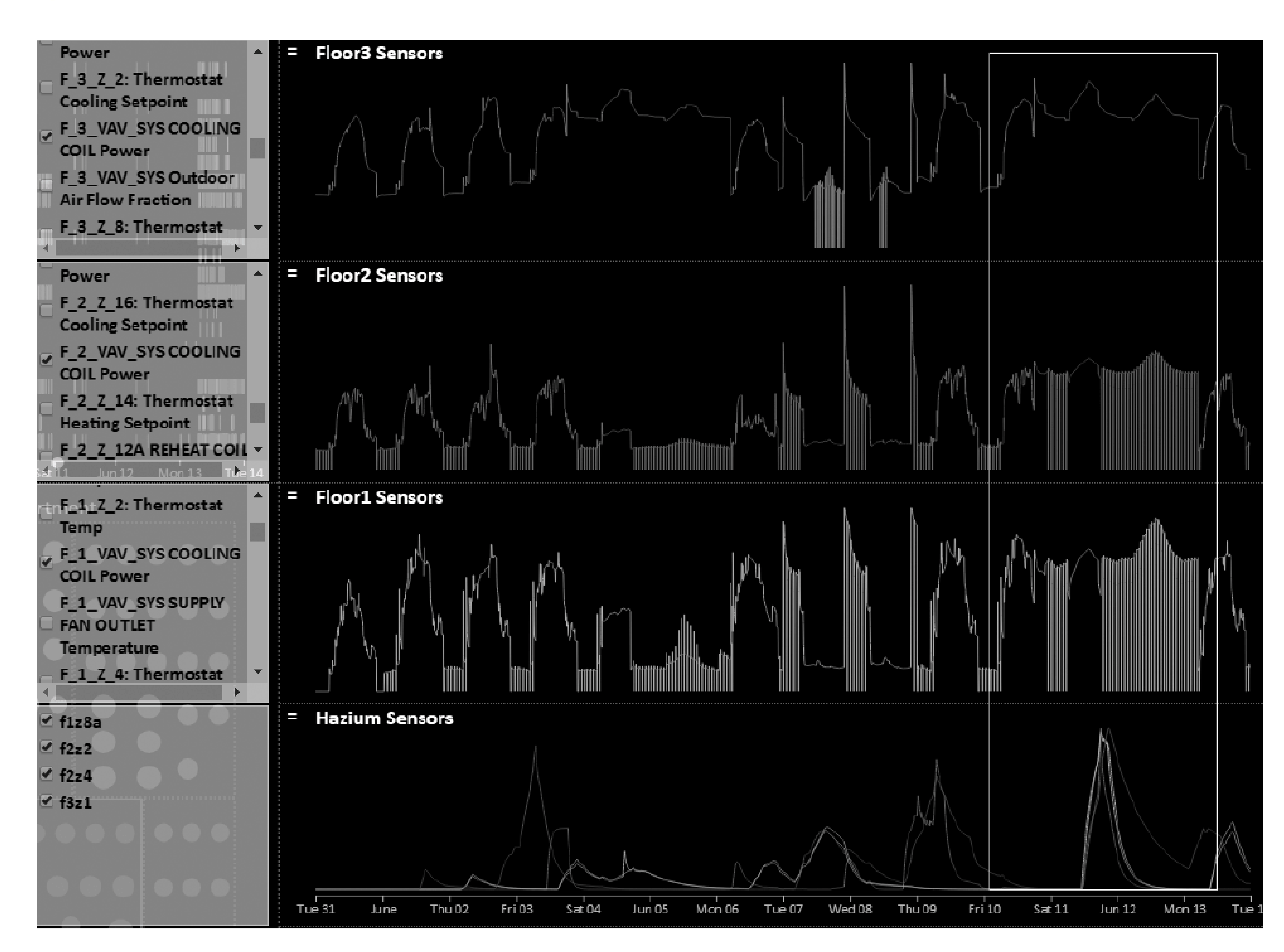


图 3-54　3 个楼层的系统冷却电源用电量和 Hazium 浓度变化

（4）描述 5 种 prox 卡和其他数据之间的关系，并分析可能的因果关系。

从图 3-55 中可以发现三层的区域 5 没有人员活动，经过查询设计图后发现这个区域为未开发区域。

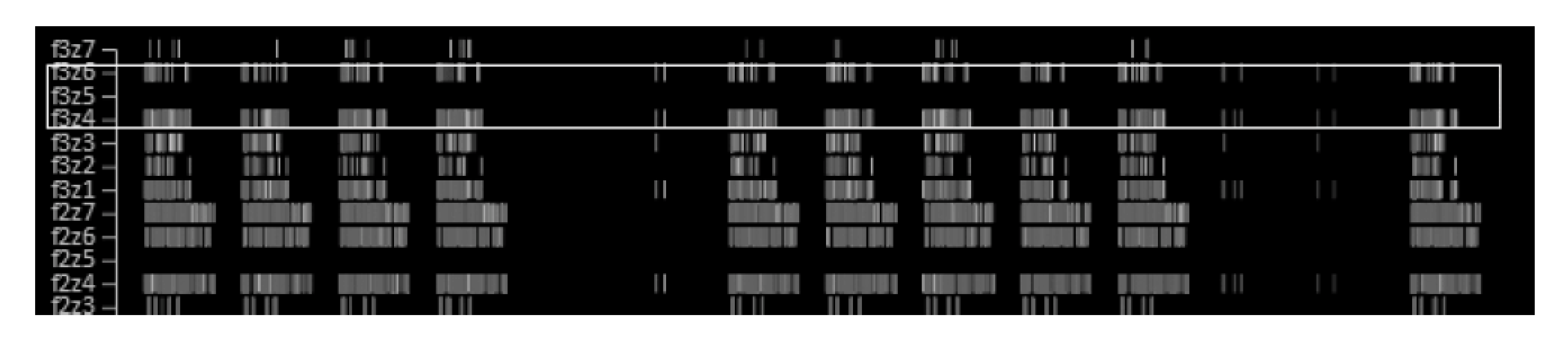


图 3-55　三层 prox 卡散点图

在图 3-56 中，发现雇员在穿越不同楼层时候基本上都会经过每层的区域 4，在查询设计图后发现每层的区域 4 为电梯或者楼梯。

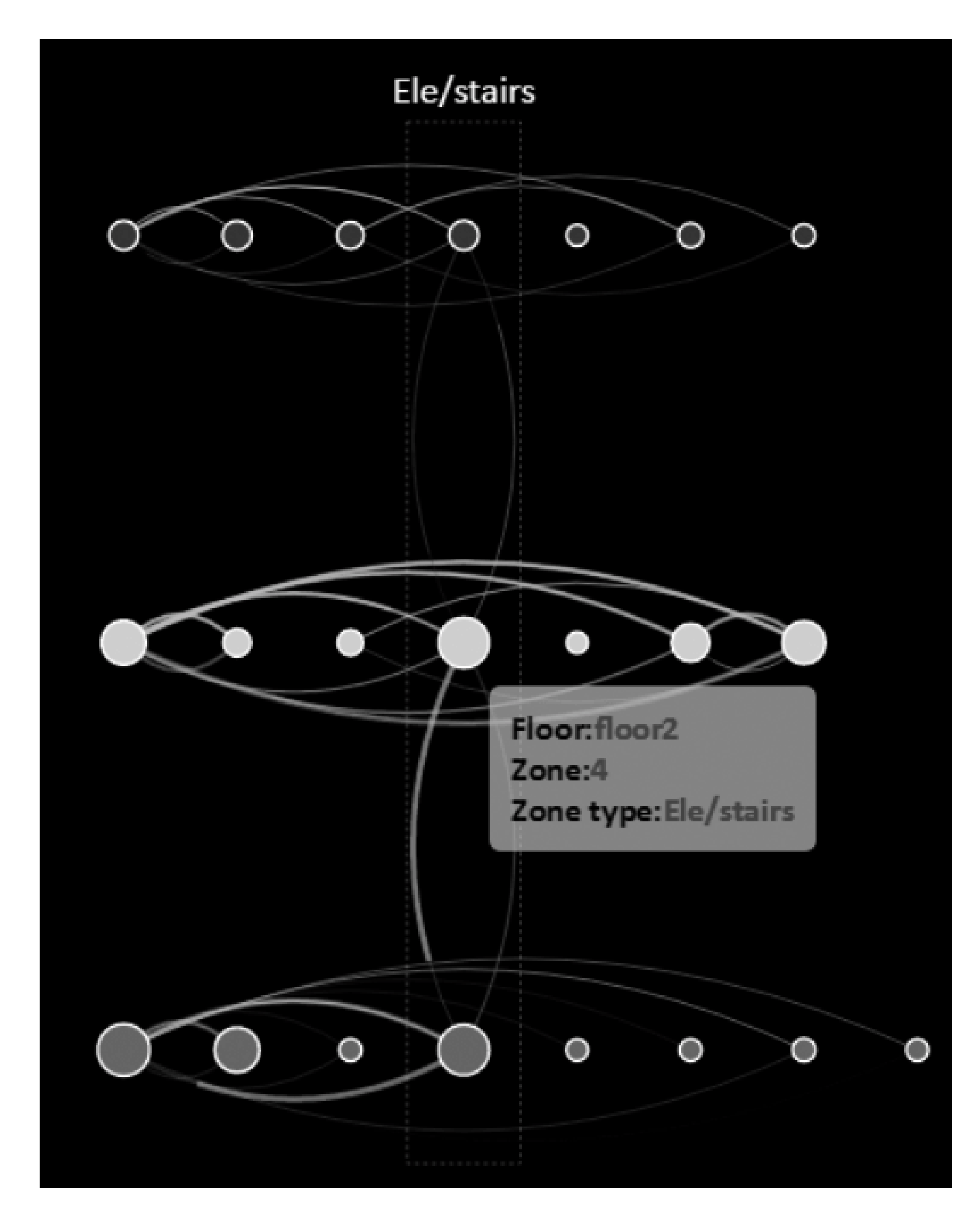


图 3-56 雇员运动轨迹图

在图 3-57 中，发现没有人员进出二层的区域 5，但是设计图上并没有这个区域的说明，因此推测这个区域为一个限制区域，一般的员工并不能进入这个区域。

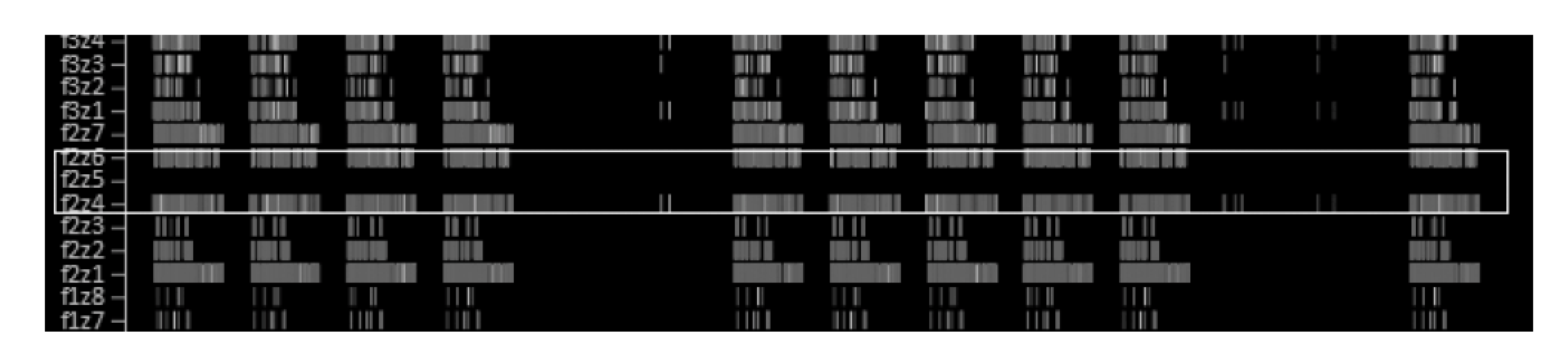


图 3-57 二层 prox 卡散点图

从图 3-58 中，发现只有两个特定的雇员出现在一层的区域 6，在查询 prox 区域设计图后发现这个区域为配置室，因此可以推测这两个雇员为管理员，只有这两个雇员才有权限进入这个区域。

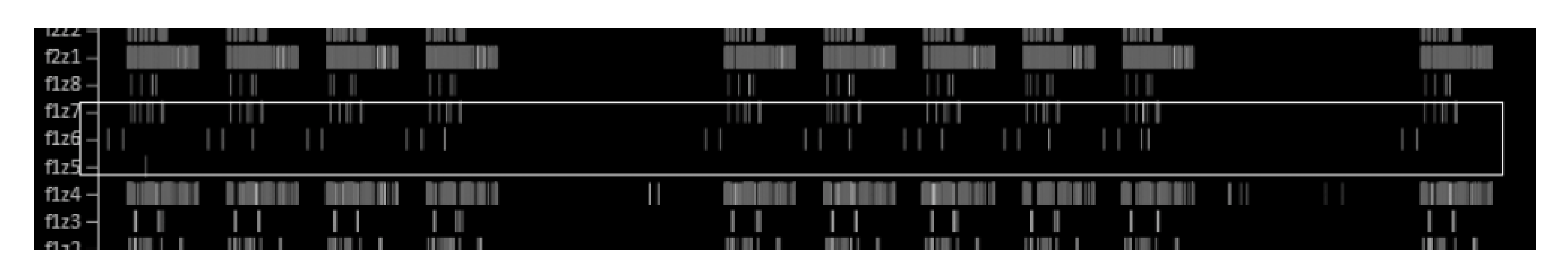


图 3-58 一层 prox 卡散点图

从图 3-59 中，在对雇员轨迹模拟后发现一层的区域 1 人流量最大，并且所有的人员在进入建筑都会经过这个区域，而这个区域为进出建筑的唯一通道。

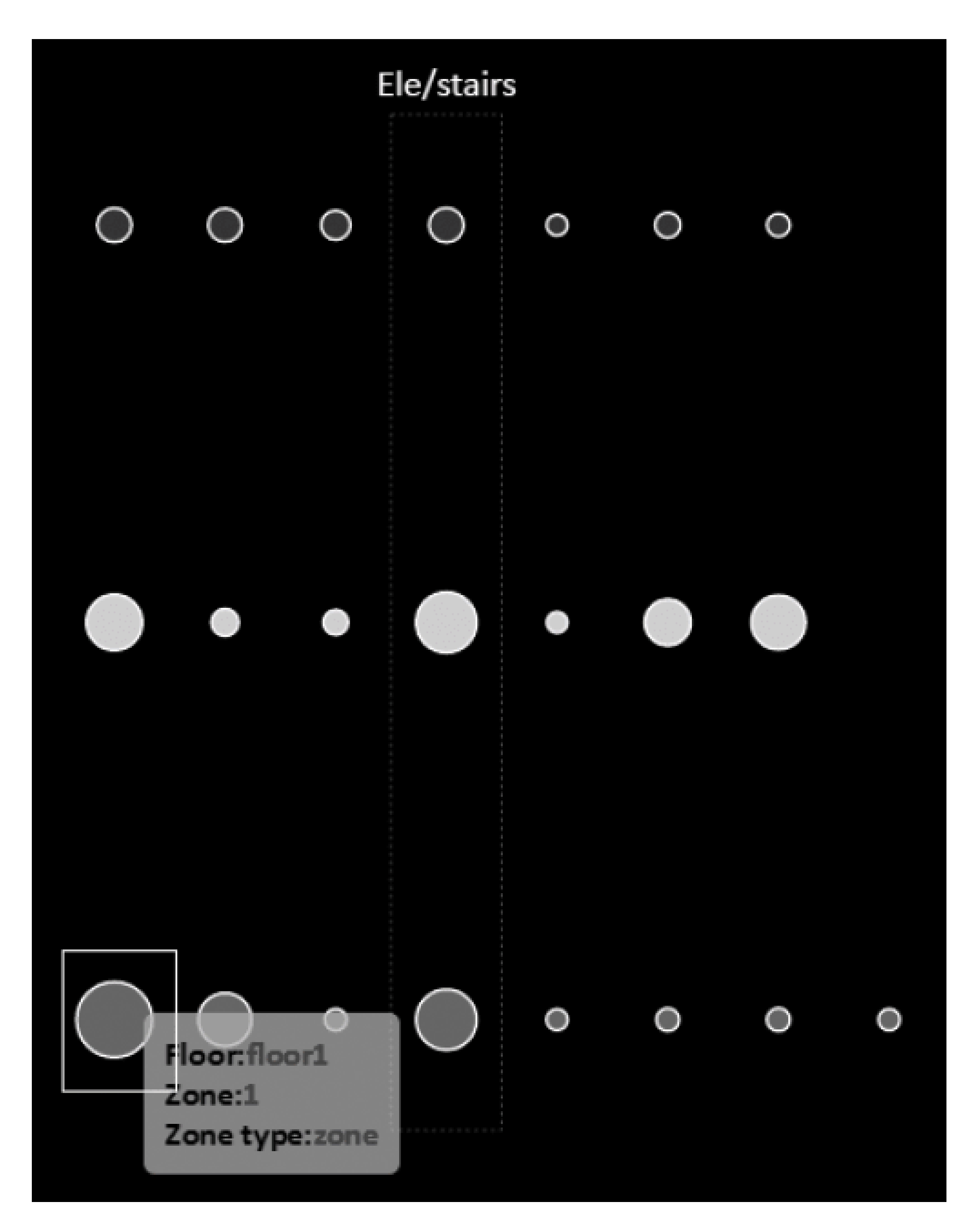


图 3-59　雇员运动轨迹图

3.5　小　　结

本章从可视化的概念出发，论述了可视化技术的分类与可视化效果的评判标准；针对大数据发展的需求，特别选择了两类数据类型的可视化：**高维数据可视化**和**网络数据可视化**。

高维数据可视化中介绍了两种常见的降维处理方法：主成分分析(PrincipleComponent Analysis，PCA)和多维尺度分析(MultiDimensional Scaling，MDS)；以及常见的高维数据可视化方法：星型图(Star Plots)、切尔诺夫脸谱图(Chernoff Faces)、散点图矩阵(Scatterplot Matrix)和平行坐标(Parallel Coordinates)。

网络数据可视化中介绍了最常用的节点—链接和相邻矩阵两种网络布局方法以及使用混合布局来实现多视图模式的原则。其中节点—链接布局中对力引导(Fruchterman—Reingold)布局、多维尺度分析(Multidimensional Scaling)布局以及弧长链接图(Arc Diagram)进行了详细描述。

最后，在此基础上介绍了两个具体的可视化实例，学生可以借助实例建立自己的可视化方案，从而达到从理论到实践的提升。

3.6 习　　题

1. 用 MATLAB 实现 PCA。

2. 只知道空间中的点(样本)的距离,那么怎么来重构这些点的相对位置呢?(提示:用 MDS 方法解决。)

3. 根据“2014 年美国各州的犯罪率”数据画星形图或切尔诺夫脸谱图(Chernoff Faces)。(数据下载网址:http://www.ojjdp.gov/ojstatbb/crime/JAR.asp。)

4. 以鸢尾花数据为例,画散点图矩阵或平行坐标。(数据下载网址:http://wenku.baidu.com/link?url=_Lup8DTSRSfMZnZPI5kPCgAoeSQYYsTDn9EdNVENFzSOfI_fIwjiYVu8rAV3pAmuPawieMk4HX630qKwPycQ6m4hTslW1cbmbxjAlrcdELHa。)

5. 对高维数据可视化的方法进行优缺点总结。

6. 采用多种算法对 MNIST 数据集降维和可视化,并对实验结果作出说明。

7. 已知中国几大城市之间的距离,但是不知道它们的经纬度,求它们之间的相对位置关系。(提示:用 MDS 方法分析。)

	北京	天津	上海	重庆	呼和浩特	乌鲁木齐	拉萨	银川	南宁	哈尔滨
北京	0	125	1239	3026	480	3300	3736	1192	2373	1230
天津	125	0	1150	1954	604	3330	3740	1316	2389	1207
上海	1239	1150	0	1945	1717	3929	4157	2092	1892	2342
重庆	3006	1954	1945	0	1847	3202	2457	1570	993	3156
呼和浩特	480	604	1717	1847	0	2825	3260	716	2657	1710
乌鲁木齐	3300	3330	3929	3202	2825	0	2668	2111	4279	4531
拉萨	3736	3740	4157	2457	3260	2668	0	2547	3431	4967
银川	1192	1316	2092	1570	716	2111	2547	0	2673	2422
南宁	2373	2389	1892	993	2657	4279	3431	2673	0	3592
哈尔滨	1230	1207	2342	3156	1710	4531	4967	2422	3592	0

8. 利用学习到的可视化方法,任选两种实现,可以是上面的弧长链接法、相邻矩阵法、力引导法或者其他方法,例如邻接图、矩阵树图、气泡图。

9. 比较混合布局的 MatrixExplorer 和 NodeTrix 两种方法,分析它们的优缺点。

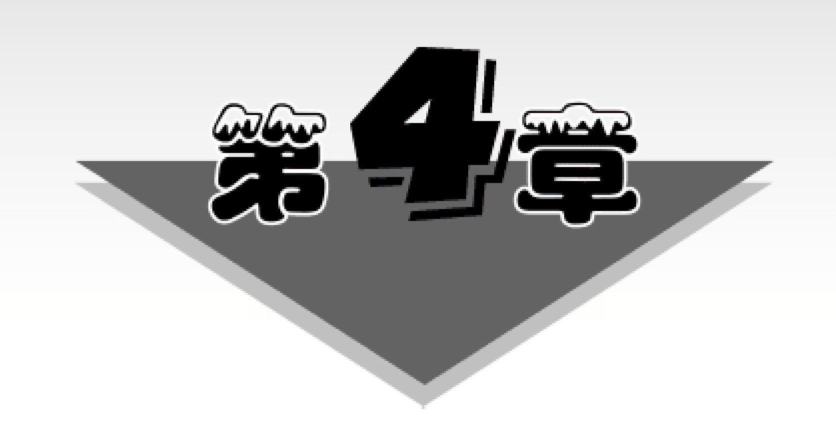
10. 根据本章的案例分析,自己做一个小型案例分析系统。

3.7 参 考 文 献

[1] 袁晓如,郭翰琦,肖何,等. 高维数据可视化. 中国计算机学会通信,2011,7(4):13-16.

[2] Elzen S, Holten D, Blaas J, et al. Reducing Snapshots to Points: A Visual Analytics Approach to

Dynamic Network Exploration[D/OL]. ISSN：1077-2626.[2016-1].
https：//www.computer.org/csdl/trans/tg/2016/01/07192717-abs.html.
[3] Alpaydin E. 机器学习导论. 范明,昝红英,牛常勇,译. 北京：机械工业出版社,2014.
[4] 陈为,沈则潜,陶煜波. 数据可视化. 北京：电子工业出版社,2013.
[5] 赵守盈，吕红云. 多维尺度分析技术的特点及几个基础问题. 中国考试，2010,(4)：13-19.
[6] 吕之华. 精通 D3.js：交互式数据可视化高级编程. 北京：电子工业出版社,2015.



数据关联分析

本章主要关注在大数据分析中数据对象之间的关联分析方法，从而发现大数据下对象之间的隐含关系及相互影响，确定是否存在一(多)事件的发生，引发了另一(多)事物的反应等。通过这种关联分析，能够更好地发现这些现象的本质，更好地掌握事物的动态发展趋势。关联分析技术已经被广泛应用在金融行业，它可以用于预测银行客户需求。通过应用这些信息，银行可以改善自身营销方案。比如各银行可以在自己的 ATM 机上捆绑顾客感兴趣的本行产品信息，供使用本行 ATM 机的用户了解。同样，如果数据库中显示某个客户更换了常住地址，说明这个客户很有可能新近购买了一栋更大的住宅，因此该客户可能需要信用额度更高的信用卡，或者需要一个住房改善贷款，这些产品信息都可以通过信用卡账单邮寄给客户。同样当该客户打电话咨询的时候，在数据库中显示的相关信息可以有力地帮助电话销售代表进行产品营销。随着互联网经济的发展，一些知名的购物网站也从强大的关联分析方法中受益。这些购物网站使用关联分析方法进行规则挖掘，可以分析出用户有意要一起购买的捆绑包。也有一些购物网站使用这些规则设置相应的交叉销售，这就是网上购物时经常会在所购商品网页上看见相关商品的广告。

关联分析方法中很多算法都有着成熟的应用，正是由于关联规则算法的实际应用推动了机器学习算法的发展，也对数据分析研究领域产生了巨大的影响。本章将从一个问题案例出发(4.1 节)，然后介绍关联规则分析中的基本概念和术语(4.2 节)。在 4.3 节，将对 3 种典型的频繁项集挖掘算法进行学习，了解关联规则挖掘的整个方法过程。针对大数据集的出现，为了提高关联分析算法的效率，许多学者提出了一些十分有效的处理大数据集的数据结构，在 4.4 节将对几种典型的数据结构进行分析。然而在实际应用中，并不是所有挖掘到的关联规则都是有效的，如何评估它们的可用性和有效性将成为关联规则分析中的一个重要研究内容，在 4.5 节将讨论几种典型的评估指标和计算方法。接下来将把讨论扩展到频繁项集挖掘的一些高级方法，比如多维关联规则挖掘(4.6 节)以及多层关联规则挖掘(4.7 节)。最后使用 Python 语言给出了一个经典的 Apriori 算法应用案例的全过程(4.8 节)。

4.1 数据关联分析简介

为了便于概念理解，下面将通过一个超市的销售案例介绍关联规则算法的应用。

例 4-1 某大型超市为了能够科学合理地重新布置货架和物品，根据顾客的购买事

务数据进行了购物篮分析，有效地解决了问题。商场的原有布局模式比较单一，例如，食品区域只是简单地分为食品和饮料。货架和物品的原有摆放如图 4-1 所示。

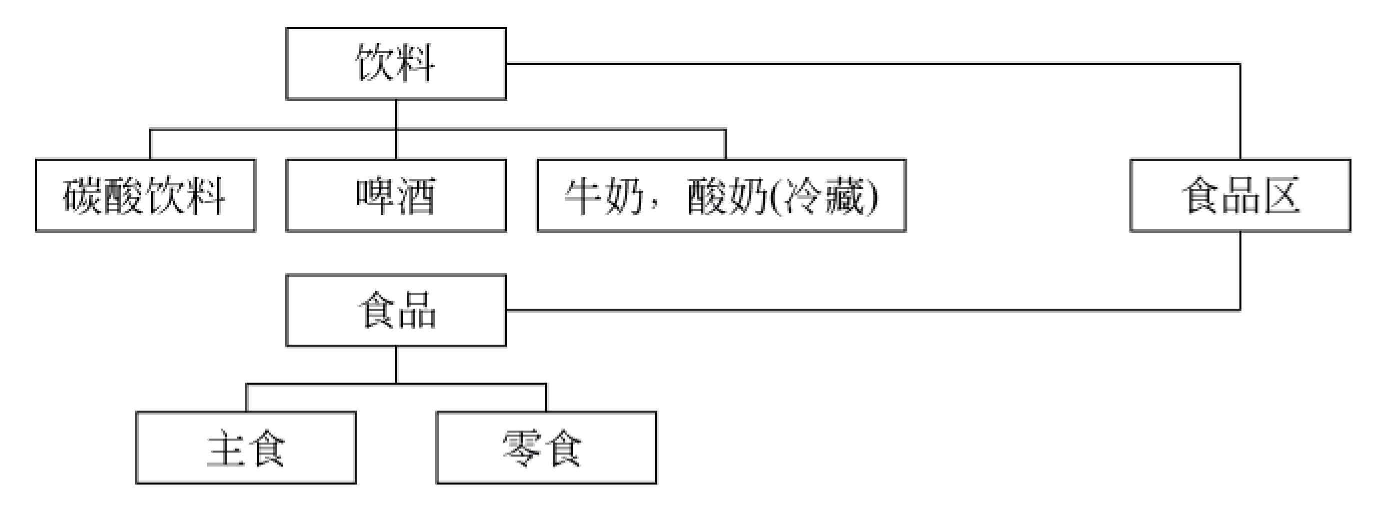


图 4-1　商品的原有布局

顾客购物的交易数据如表 4-1 所示，为简单描述关联分析算法应用的情况，在这里只抽取 10 条数据和 6 个特征进行展示。

表 4-1　顾客购物交易数据

订单号	啤酒	碳酸饮料	主食	牛奶	酸奶	零食
08012101	1	0	1	1	0	1
08012102	0	1	0	0	1	1
08012103	1	0	1	1	0	0
08012104	0	1	0	0	0	1
08012105	1	0	1	0	0	1
08012106	0	1	0	0	1	0
08012107	0	1	0	0	0	1
08012108	1	0	0	0	0	1
08012109	1	1	1	1	0	0
08012110	0	1	0	0	1	1

在数据表中，第一行为商品名称，从第二行开始每一行代表一个购物记录事务。表中的 1 为购买，0 为没有购买。

在 Weka 软件平台上，通过调用 Apriori 算法，对表 4-1 中的数据进行关联规则分析，导入相关数据，调节 Apriori 算法的相关参数，然后运行得到计算结果。在最小支持度为 0.45，最小置信度为 0.9 的情况下，得到了 9 条频繁 1 项集、17 条频繁 2 项集、13 条频繁 3 项集、5 条频繁 4 项集、1 条频繁 5 项集。为了让读者能够理解分析的过程，在这里对与啤酒相关的前 5 条计算结果进行分析，相关的分析数据如下所示：

(1) 啤酒＝yes 5→酸奶＝no 5　　　conf：(1)

(2) 啤酒＝no 5→碳酸饮料＝yes 5　　　conf：(1)

(3) 啤酒＝no 5→主食＝no 5　　　conf：(1)

(4) 啤酒＝no 5→牛奶＝no 5　　　conf：(1)

(5) 主食=yes 4→啤酒=yes 4　　　　　　conf：(1)

根据以上计算的关联规则,可以得到如下分析结论：

- 顾客在购买啤酒的情况下,基本不会购买酸奶(数据(1))。
- 顾客在未购买啤酒的情况下,有很大可能会购买碳酸饮料(数据(2))。
- 顾客在未购买啤酒的情况下,基本不会购买主食或酸奶(数据(3)、(4))。
- 顾客在购买主食的情况下,会购买啤酒(数据(5))。

通过分析,得到的结论如下：

- 啤酒和酸奶不需陈列在一起,两者之间不会起到互相促进销售的作用。
- 啤酒和碳酸饮料也不需陈列在一起。
- 啤酒似乎可以和主食、牛奶陈列一起,但从上述规则不能确定。
- 啤酒和零食的规则无法得到。

通过结合其他数据的计算结果,得到能够相互促进购买的商品组合如下：

- 啤酒↔主食。
- 主食↔牛奶。
- 碳酸饮料↔酸奶、零食。

综上,案例4-1得到的超市重新布置货架和商品的情况如图4-2所示。

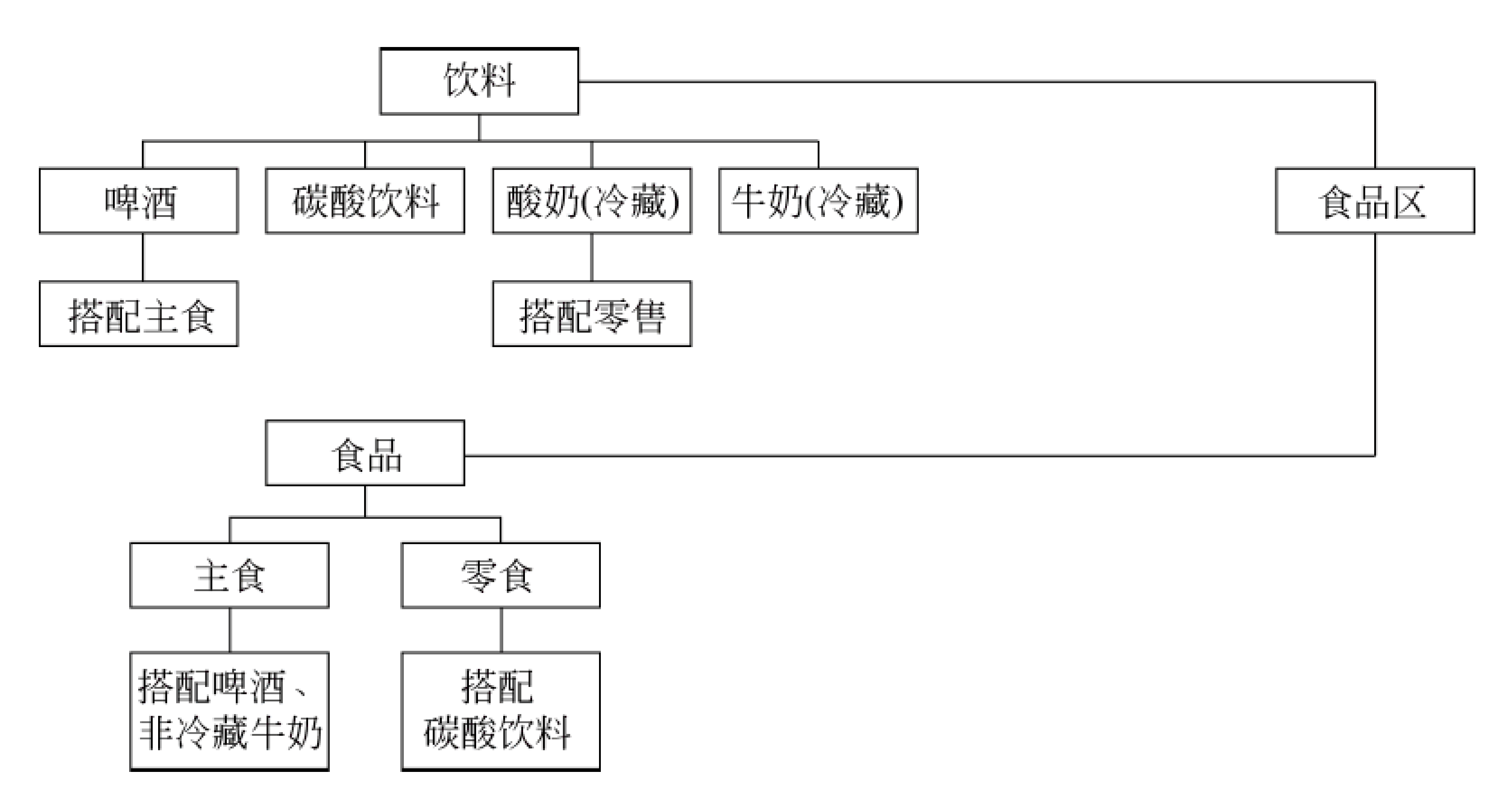


图4-2　重新布局的商品

4.2 基本概念

众多的模式分析算法,如聚类、分类归纳、决策树等,都被广泛用于数据分析应用中,它们都来源于机器学习领域。但是,频繁模式和关联规则广泛和成熟的应用为数据分析的研究和机器学习的发展起到了极大的推动作用。

频繁模式可以进一步描述成数据对象间的关联规则。如何判断序列模式是否频繁,通常使用两个基本的度量：支持度(support)和置信度(confidence)。支持度表示该序列模式在所有被考察的数据对象中的占比,也就是体现了该模式的有用性;置信度表示由模式的前因推出后果的可信度,体现了规则的确定性。由频繁模式可以产生相应的关联规

则，在满足一定条件下，这些推出的关联规则被认为是有趣和有用的，有用的关联规则可以被用于为决策层和高层管理人员提供制定商品营销方式的依据。

本节首先介绍频繁项集和关联规则中的基本概念和计算公式。在频繁项集计算中，往往会获得庞大的频繁项集，实际使用中通常只需要求出闭项集或极大频繁项集，相关概念将在 4.2.2 节中给出。在关联规则的实际应用中，人们通常不仅对数据集的频繁模式感兴趣，有时候还会对不频繁的稀有模式或负模式感兴趣，这些概念定义将在 4.2.3 节介绍。

4.2.1 频繁项集和关联规则

假设 $I=\{I_1,I_2,\cdots,I_m\}$是项的集合。给定一个交易数据库 D，其中每次交易事务 T (Transaction)是 I 的非空子集，即 $T\subseteq I$，每个 T 都与一个唯一的标识符 TID (Transaction ID)对应。

定义 4-1 项集(itemset)。是指项的集合。包含 k 个项的项集称为 k 项集。例如集合{computer，printer}就是一个 2 项集。项集的出现频数是包含项集的事务数，简称为项集的频数、支持度或计数。

定义 4-2 支持度(support)。简单的字面理解就是支持的程度，一般以百分比表示。设 A 是一个项集，在数据挖掘的关联分析中，支持度表示在所有的事务中同时出现 A 项(或项集)的概率，计算公式如下：

$$\text{support}(A)=\frac{\text{count}(A\subseteq T)}{|D|} \tag{4-1}$$

支持度也可以通过计算项集在数据集 D 中出现的次数来考察其频数，这时也可以称为支持度计数。

定义 4-3 频繁项集(frequent itemset)。在一个具体的数据分析任务中，用户或者领域专家可以自行设定最小支持度阈值，那么如果项集 A 的支持度满足预定义的最小支持度阈值(即 A 的支持度不小于最小支持度阈值)，则 A 是频繁项集。如果一个包含 k 个项的项集的支持度大于等于最小支持度阈值，则该项集称为 k 频繁集。

定义 4-4 关联规则(Association Rule)。是有关联的规则，可以描述成一种蕴含式，比如 $A\rightarrow B$，其中 A 和 B 分别称为关联规则的前件(antecedent 或 Left-Hand-Side，LHS)和后件(consequent 或 Right-Hand-Side，RHS)。例如，在某超市分析中发现，购买面包的顾客一般会购买牛奶，则{面包}→{牛奶}就是一条关联规则。关联规则从一个侧面揭示了事务之间的某种联系。支持度和置信度总是伴随着关联规则存在的，它们是对关联规则的必要补充。关联规则支持度的定义与频繁集支持度的定义基本相同，只有少许区别。在事务集合 D 中，对某条关联规则而言，其支持度 s 表示在所有的事务中同时出现A 和 B 的概率，即 $P(AB)$，其计算公式如下：

$$\text{support}(A\Rightarrow B)=\frac{\text{count}(A\cup B)}{|D|} \tag{4-2}$$

定义 4-5 置信度(confidence)。又称为可信度，它表示关联规则的前件出现时后件是否也会出现，如果出现，那么出现的概率有多大。在事务集合 D 中，对某条关联规则 $A\rightarrow B$ 而言，置信度表示 A 出现时 B 同时出现的概率，即 $P(B|A)$。例如，在某超市分析

中,发现置信度为100%,则A和B可以捆绑销售置信度的。如果置信度太低,则说明A的出现与B是否出现关系不大。置信度的计算公式如下:

$$\text{confidence}(A \Rightarrow B) = \frac{\text{support}(A \cup B)}{\text{support}(A)} \tag{4-3}$$

同样,在一个具体的数据分析任务中,用户或者领域专家可以自行设定最小支持度阈值和最小置信度阈值,如果某个关联规则同时满足最小支持度阈值(min_sup)和最小置信度阈值(min_conf),则认为这个关联规则是有趣的。

定义 4-6　强关联规则。如果某关联规则$A \rightarrow B$满足$\text{support}(A \Rightarrow B) \geqslant \text{min_sup}$且$\text{confidence}(A \Rightarrow B) \geqslant \text{min_conf}$,则称该关联规则是强关联规则,否则称为弱关联规则。

例 4-2　支持度和置信度分析。

为了能够更加详细地阐述支持度和置信度,下面以运动商品的购买交易数据为例进行分析。事务数据如表4-2所示。

表 4-2　购买运动商品交易数据

TID	网球拍	网 球	运动鞋	羽毛球
1	1	1	1	0
2	1	1	0	0
3	1	0	0	0
4	1	0	1	0
5	0	1	1	1
6	1	1	0	0

以上为运动商品购买事务数据的前6条数据和4个特征,第一行为商品名称和事务编号,第一列为事务编号,每个事务购买记录1表示购买,0表示未购买。项集$I=\{$网球拍,网球,运动鞋,羽毛球$\}$。仅考虑频繁2项集情况,讨论网球拍和运动鞋的关联规则关系,如果设定最小支持度阈值$\alpha=0.4$,最小置信度阈值$\beta=0.4$,在事务记录中,第1、2、3、4、6个事务包含了网球拍,第1、4、5个事务包含了运动鞋。令包含网球拍的事务集为x,包含运动鞋的事务集为y,则$x \wedge y=2$,整个事务集合$|D|=6$,于是关联规则"网球拍→运动鞋"的支持度和置信度计算如下:

支持度:$(x \wedge y)/D=0.33$(同时包含x和y的概率)。

置信度:$(x \wedge y)/x=0.40$(在x条件下含有y的概率)。

由此关联规则"网球拍→运动鞋"不满足强关联规则的定义。

同理,分析网球拍和网球的关联关系,在事务记录中,第1、2、3、4、6个事务记录包含了网球拍,第1、2、5、6个事务记录包含了网球,因此网球和网球拍为频繁2项集。根据支持度和置信度的计算规则,可以得到支持度为0.5,置信度为0.6,则关联规则"网球拍→网球"存在关联,是强关联规则。

一般只有支持度和置信度均较高的规则才是用户感兴趣的,这也是关联规则的核心所在。

关联规则的挖掘主要有两个关键步骤：

(1) 从数据集合中找出所有的频繁项集。从数据集合中找到的项目集合的支持度必须大于等于所设定的最小支持度阈值。通常是先找出频繁 1 项集，然后从频繁 1 项集找出频繁 2 项集，以此类推，进一步从频繁 k 项集中再产生频繁 $k+1$ 项集，直到无法再找到更长的频繁项集为止。

(2) 从找到的所有长度大于 2 的频繁 k 项集中产生关联规则。利用步骤(1)产生的长度大于等于 2 的频繁项集来产生规则，若一规则所求得的置信度满足最小置信度阈值，称此规则为强关联规则。

4.2.2 闭项集和极大频繁项集

从庞大数据集中挖掘频繁项集常常出现的问题是产生大量满足最小支持度阈值的项集，当最小支持度阈值(min_sup)设置得很低时更是如此。主要是因为一个项集是频繁的，则它的每一个子集都是频繁项集，一个长项集合可以包含了组合个数的频繁子集。例如，一个长度为 100 的频繁项集$\{a_1, a_2, \cdots, a_{100}\}$包含了 100 个频繁 1 项集和 4950 个频繁 2 项集$\{a_1, a_2\}$，$\{a_1, a_3\}$，…，$\{a_1, a_{100}\}$，$\{a_2, a_3\}$，$\{a_2, a_4\}$，…，$\{a_2, a_{100}\}$，…，$\{a_{99}, a_{100}\}$，以此类推。这样，频繁项集的总个数约为 1.27×10^{30}。可见，如果要获取关联规则，项集个数越多，意味着关联规则分析所要花费的时间和空间越大，对于计算机的计算能力和存储都提出了更高的要求。而实际应用中，并不需要分析所有满足条件的频繁项集和关联规则，因此，为了让分析具有一般性，提出了闭频繁项集和极大频繁项集等概念。

定义 4-7　超项集(super itemset)。若一个集合 S_2 中的每一个元素都在集合 S_1 中，且集合 S_1 中可能包含 S_2 中没有的元素，则集合 S_1 就是 S_2 的一个超项集。若 S_1 是 S_2 的超项集，则 S_2 是 S_1 的子集。

定义 4-8　闭频繁集(closed frequent itemset)。在事务数据库 D 中，对于一个项集 X，它的真超项集 Y 的支持度计数不等于它本身的支持度计数，则它称为数据集 D 的闭项集。如果闭项集 X 在数据集 D 中同时也是频繁的，也就是它的支持度大于等于最小支持度阈值，那它也称为 D 中的闭频繁项集。

定义 4-9　极大频繁集(maximal frequent itemset)。如果 X 是数据集 D 中的一个频繁项集，而且 X 的任意一个超项集 Y 都是非频繁的，则称 X 是数据集 D 中的极大频繁项集，也就是说，如果 X 这个频繁项集进一步扩充就不是频繁集了。

从以上定义看出，3 种频繁项集间存在如下的关系：极大频繁集＜闭频繁集＜频繁项集。不过在实际应用中，极大频繁集会丢失很多信息。例如在超市商品销售分析中，可能在同时购买酒、花生和饼干的人群中还有一部分同时也购买了洗发水，其项集的支持度也达到了最小支持度阈值，那么项集{酒、花生、饼干、洗发水}就是项集{酒、花生、饼干}的一个超项集，这样项集{酒、花生、饼干}这个食品集合的独特性就不会在极大频繁集里出现；而闭频繁集就能够保留这类独特食品频繁项集的信息，可以继续被拆分为频繁项集来获得有用的关联规则。

例 4-3　闭频繁集和极大频繁集分析。

为了更清晰地了解闭项集和极大频繁项集的概念，下面对一个实例进行分析，数据信

息如表 4-3 所示。

表 4-3　项集列表

项集	支持度	极大频繁集	闭频繁集
A	4	NO	NO
B	5	NO	YES
C	3	NO	NO
AB	4	YES	YES
AC	2	NO	NO
BC	3	YES	YES
ABC	2	NO	NO

这里定义最小支持度为 3，则 A、B、C、AB、BC 支持度都大于或等于 3，因此它们为频繁项集，其中 AB 为 A、B 的并集，其他并集以此类推。

首先理解极大频繁项集的概念。如果 X 是数据集 D 中的一个频繁项集，而且 X 的任意一个超项集 Y 都是非频繁的，则称 X 是数据集 D 中的极大频繁项集，也就是说，如果 X 这个频繁项集进一步扩充就不是频繁集了。观察表 4-3 可知，A、B 是频繁项集，它们的组合超项集 AB 也是频繁的，这时看 AB 的超项集 ABC，由表可得它不是频繁的，所以 AB 为极大频繁项集。同理，可以得到 BC 也为极大频繁项集。

然后理解闭频繁项集的概念。在事务数据库 D 中，对于一个项集 X，它的真超项集 Y 的支持度计数不等于它本身的支持度计数，则它称为数据集 D 的闭项集。如果闭项集 X 在数据集 D 中同时也是频繁的，也就是它的支持度大于等于最小支持度阈值，那它也称为 D 中的闭频繁项集。观察表 4-3 可知，A 是频繁项集，它的超项集 AB 的支持度等于 A 的支持度，所以 A 不是闭项集。但是项集 AB 支持度大于它的超项集 ABC 支持度，则可以得到 AB 为闭项集。同理，可以得到 B 和 BC 为闭项集。

4.2.3　稀有模式和负模式

以上介绍的都是与挖掘频繁模式相关的概念，然而在数据分析中，不同的应用需求需要也必须具备多种求解复杂问题的思维模式。例如，在饰品中宝石是稀有的，在服饰中奢侈品也总是少数人才能够拥有的，对这类商品间的关联分析挖掘也通常是最让人感兴趣的。同样，在超市商品销售分析中，购买可口可乐的顾客一般不会购买百事可乐，而顾客购买了这两种商品可能就是负相关事件，发现和观察这种异常现象对数据挖掘同样起着重要作用，以上的数据关联分析称为稀有模式分析和负模式的分析，因此提出了稀有模式和负模式的概念。

定义 4-10　稀有模式(rare mode)。又称为非频繁模式。给定一个用户指定的正常数 δ，其满足 $0<\delta<1$，则该类模式的最小支持度阈值通常定义为

$$R\text{min_sup} = \delta \times \text{min_sup}$$

其中 min_sup 为频繁项集的最小支持度阈值，若一个模式的支持度低于(或远低于)该最

小支持度阈值，这个模式就称为稀有模式。

现有的数据关联分析基本都是关注频繁模式的挖掘，希望通过这种模式的挖掘揭示出数据集所蕴含的规律。然而在一些应用领域，数据集中不频繁出现的序列反而会揭示出人们未知的规律。例如在金融安全领域，相比于普通的交易行为，银行管理人员更关注非正常的交易行为，比如欺诈交易等，但这种欺诈行为也不是随机出现的，它们也具有一些共性的特征，因此可以采用数据分析中的稀有模式挖掘方法发现关联规则，进而揭示出非正常的稀有行为的特点。

定义 4-11　负模式(negative mode)。在数据集 D 中，某数据项集 X 的出现会减少另一数据项集 Y 的出现，甚至使得 Y 不出现，这一类模式称为负模式，由此产生的关联规则称为负关联规则。例如，在超市商品销售中，分别购买可口可乐和百事可乐的支持度都非常高，但是二者存在很强的负相关性，因此它们可以组成负数据项集，由此而生成的关联规则称为负关联规则。

负关联规则的描述是一个形如 $X \Rightarrow Y$ 的表达式，其中 $X, Y \subset I$，$X \cap Y = \varnothing$，X 称为规则的前件，Y 称为规则的后件。

在发现负关联规则的过程中，可以利用一些正关联规则的分析技术，即将负关联规则发现分为两个步骤，一是产生所有的负数据项集，二是产生负关联规则。正如正关联规则可以为经营者管理层提供决策支持一样，负关联规则的发现也能够有助于企业营销质量的提高。负模式分析的研究目前还较少，已有的一些相关算法包括 PNSP、Negative-GSP、NSPM、PNSPM 等。

4.3　Apriori 算法

关联规则是数据分析中最有效的手段之一。从关联规则挖掘的两个步骤可以看出，分析的关键和难点都集中在频繁项集的挖掘上。通过对频繁项集的分析，可以发现在数据集中经常同时出现的相关性较大的数据对象，从而为可能的决策提供支持。目前，对频繁项集分析的研究不存在已知的多项式复杂度的算法，普遍都需要进行计算量非常大的搜索工作，因此主流的频繁项集挖掘方法都需要对搜索空间进行有效的剪枝处理，其解决问题的思路可以归纳如下：在数据集的遍历上采用自底向上、自顶向下或混合遍历的方法；在空间搜索上采用广度优先或深度优先；在频繁项集的产生上着眼于是否首先产生候选项集。不同的遍历方法和搜索策略产生了不同的算法，实践证明，有些算法适用于稀疏事务集，有些算法适用于稠密事务集，但是没有一种挖掘算法可以很好地解决所有情形下的问题，每种相对较优的算法都有它具体的适用环境。所以，有必要熟知不同算法的不同特性，掌握不同的适用环境。

本节主要介绍频繁项集挖掘的 Apriori 算法的基本概念和思想，该算法在关联分析中较为重要，是主要的算法之一。该算法采用广度优先的搜索策略，自底向上的遍历，遵循首先产生候选集进而获得频繁集的思路。Apriori 算法思想将在 4.3.1 节详细地给出，具体的算法描述在 4.3.2 节中以伪代码形式表示，在 4.3.3 节中针对 Apriori 算法存在可能产生大量的候选集以及可能需要重复扫描数据库多次的问题，详细介绍改进的 Apriori

算法，即DHP算法。

4.3.1 Apriori算法的核心思想

1993年，Agrawal等人首先提出了关联规则的概念，同时给出了相应的关联规则挖掘算法AIS，但是性能较差。1994年，他们建立了项目集格空间理论，并依据其中的两个定理，提出了著名的Apriori算法，此后诸多研究人员相继对关联规则的挖掘问题进行了大量的研究，至今Apriori仍然作为关联规则挖掘的经典算法被广泛讨论和应用，已经成为一种最有影响的挖掘关联规则频繁项集的简单适用的算法。作为最经典的关联分析算法，Apriori算法采用广度优先的搜索策略，自底向上的遍历思想，遵循首先产生候选集进而获得频繁集的思路。该算法适合用在数据集稀疏，事务宽度较小，频繁模式较短，最小支持度较高的环境中；而对于稠密数据和长频繁模式，由于候选项集占据大量的内存，计算成本急剧增加，数据集的遍历次数加大，因而该算法的性能下降。

不管是哪种频繁项集挖掘算法，都用到了频繁项集的反单调性原理。基于该原理可以对搜索空间进行有效的剪枝。

定义4-12 反单调性原理。如果一个项集是频繁的，那么它的所有子集也是频繁的，也就是说如果一个项集是非频繁的，那么它的所有超集也一定是非频繁的。

Apriori算法的核心思想是：首先扫描数据集，统计数据集中交易的数量和各个不同的1项集出现的次数，进而根据最小支持度min_sup获得所有的频繁1项集，即L_1，然后利用L_1查出找频繁2项集L_2，如此继续，直到不再有新的频繁项集被找到为止。归纳起来是两个步骤：连接步和剪枝步。

(1) **连接步**。为找L_k(频繁k项集)，通过将L_{k-1}与自身连接产生k项集，该k项集记作C_k，但是两两自连接时只能对只差最后一个项目不同的项集进行连接。

(2) **剪枝步**。显然超集中的成员有些是频繁的，有些不是频繁的。但所有的频繁k项集都包含在其中。如果扫描数据集，对其中每个成员项集进行计数，找出那些计数值不小于min_sup的成员，从而构成L_k(即频繁k项集)。但是，这种计算中的成员数量可能很大，造成计算量增大，因此算法在实际操作中使用反单调性原理：通过判断候选集的所有成员的$k-1$项集是否是频繁的，只要有一个不是$k-1$频繁项集，那么该成员不可能成为k频繁项集，从而可以将其删除，最后剩下的成员构成了被剪枝的候选集；然后再采用数据集扫描方式对其中的成员进行计数，确定计数值不小于min_sup的成员，以构成频繁项集集合。

例4-4 生成候选集方法。

为了便于理解相关概念，这里给出一个简单的例子，让读者明白候选集的构成过程。现有数据集合D，已经找到的频繁3项集如下：

$$L_3 = \{abc, abd, acd, ace, bcd\}$$

为找到频繁4项集，将通过连接步和剪枝步实现，过程如下：

(1) 连接步：$L_3 * L_3$。

由频繁3项集$\{abc\}$和$\{abd\}$得到自连接后的集合$\{abcd\}$。

同样，由频繁3项集$\{acd\}$和$\{ace\}$得到自连接后的集合$\{acde\}$。

(2) 剪枝步：分析以上得到的两个 4 项集，其中 4 项集{$acde$}的子集{cde}不在频繁 3 项集 L_3 中，显然根据反单调性原理要进行剪枝，因此候选 4 项集 $C_4=\{abcd\}$，再扫描数据库，对该候选集进行计数，判断其计数值是否大于等于最小支持度计数，进而找出频繁 4 项集。

4.3.2 Apriori 算法描述

Apriori 算法的功能是寻找所有支持度不小于 min_sup 的频繁项集。它使用一种称作逐层搜索的迭代方法。在后续的每次遍历中，利用上一次遍历所得频繁项集作为搜索项集，产生新的潜在频繁项集——候选项集，然后对候选项集进行筛选，寻找频繁项集，如此循环，直到不能再找到更长的频繁项集为止。

算法 4-1 Apriori 算法伪代码描述

输入：交易数据库 D；最小支持度 min_sup 计数阈值

输出：D 中的频繁项集 L

```
L1=find_frequent_1-itemsets(D);
for(int k=2; Lk-1 != ϕ;k++){
    Ck=apriori_gen(Lk-1, min_sup);
    for each 事务 t∈D{          //循环遍历 D 项集
        Ct=subset(Ck,t);        //获得 t 的子集
        for each 候选 c∈C
            c.count++;
}
Lk={c∈Ck|c.count≥min_sup}
}
 return L=∪k Lk;
Procedure apriori_gen(Lk-1: frequent(k-1)-itemsets; min_sup: minimum support threshold)
for each 项集 l1∈Lk-1
    for each 项集 l2∈Lk-2
        if((l1[1]=l2[1])∧(l1[2]=l2[2])∧…∧(l1[k-2]=l2[k-2])∧(l1[k-1]<l2[k-1])then{
            c=l1∞l2;            //连接步，产生候选
            if has_infrequent_subset(c, Lk-1)then
                delete c;       //剪枝步，删除子集不是频繁集的候选集
            else add c toCk;}
return Ck;
Procedure has_infrequent_subset(c: candidate k-itemset; Lk-1: frequent(k-1)-itemset)
                                                              //利用原数据集
for each (k-1)-subset s of c
    if s ∉ Lk-1 then
    return true;
return false;
```

例 4-5 Apriori 算法实例分析。该数据来源于 weka 购物篮测试数据，选取其中的 10 条购物篮事务数据 D 进行说明，如表 4-4 所示。为了便于理解和计算，用 I1、I2 等编号表示商品。这里最小支持度为 2，即 min_sup=2。

表 4-4 购物篮抽取事务数据

购物篮 TID	商品 ID 列表	购物篮 TID	商品 ID 列表
T001	I2，I3，I11	T006	I1，I9，I5
T002	I2，I11	T007	I7，I5
T003	I4，I6，I7，I10	T008	I1，I6，I9
T004	I3，I8	T009	I1，I10
T005	I2，I8，I10	T010	I1，I2，I3，I4，I8，I10

在第一次扫描整个数据库后，可以确定频繁 1 项集 L_1={I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11}，每一项的支持度都大于等于最小支持度。再从 L_1 中产生 2 项集 L_2={{I1 I2}{I1 I3}…{I1 I11}{I2 I3}{I2 I4}…{I2 I11}…{I10 I11}}，比较每个 2 项集支持度与最小支持度的大小，得到频繁 2 项集为{{I1 I9}{I1 I10}{I2 I3}{I2 I8}{I3 I8}{I4 I10}{I8 I10}}。其他频繁项集以此方法类推，直到再也不能寻找到更长的频繁项集为止。流程如图 4-3 所示。

4.3.3 改进的 Apriori 算法

从以上对 Apriori 算法的介绍可以看出来，其两大缺点是可能产生大量的候选集，以及可能需要重复扫描数据库多次。针对 Apriori 算法的不足，人们提出了各种优化方法，主要包括 DHP(Direct Hashing and Pruning)算法、Partition 算法、Sample 算法和 DIC(Dynamic Itemsets Counting)算法等。由于 Apriori 算法开销集中在从数据集中提取有效信息的初始阶段，因此 DHP 算法采用哈希表对 2 项集进行优化，在 k 频繁项集产生过程中，将可能的 $k+1$ 候选项集散列到哈希表中，于是在下一步由 k 频繁项集产生 $k+1$ 项候选集时，使用前一阶段建立的哈希表过滤非频繁项集，从而减少需要在数据库扫描阶段进行支持度计数的候选项集数目。该算法还有一个特点是能够逐步减少数据集的大小，包括其中事务的宽度和数目。考虑到 Apriori 算法对数据集的扫描次数和扫描效率的瓶颈问题，Partition 算法采用了分治的思想，将数据集在逻辑上分成几个不重叠的分区，确定分区大小的原则是保证每个分区能够被放入内存，算法的核心思想是全局频繁项集至少在一个分区中应该是频繁项集。该算法总共只需要扫描数据集两次，第一次是产生、合并各个分区各自的频繁项集，第二次是验证全局频繁项集。但是该算法受到数据集不平衡情况以及占用内存不好估计的影响，在使用上受到一定的限制。Sample 算法使用了抽样的思想，核心是抽取事务数据形成可以放入内存大小的数据集，然后对此抽样数据集进行频繁项集的挖掘。该算法虽然减少了数据集的大小，降低了扫描等分析数据的计算量，但是容易漏掉一些频繁项集，对结果有一定影响。DIC 算法从减少扫描次数入手，将原数据集在逻辑上进行划分，循环扫描各个数据划分，因此主要还是集中在优化 I/O 开销上，

扫描数据库D，对每个候选集计数

候选项集	支持度计数
{I1}	4
{I2}	4
{I3}	3
{I4}	2
{I5}	2
{I6}	2
{I7}	2
{I8}	3
{I9}	2
{I10}	4
{I11}	2

产生频繁集

频数项集	支持度计数
{I1}	4
{I2}	4
{I3}	3
{I4}	2
{I5}	2
{I6}	2
{I7}	2
{I8}	3
{I9}	2
{I10}	4
{I11}	2

得到候选2项集

候选项集	支持度计数
{I1, I2}	1
{I1, I3}	1
{I1, I4}	1
{I1, I5}	1
⋮	
{I9, I11}	0
{I10, I11}	0

产生频繁集

候选项集	支持度计数
{I1, I9}	2
{I1, I10}	2
{I2, I3}	2
{I2, I8}	2
{I3, I8}	2
{I4, I10}	2
{I8, I10}	2

得到候选3项集

候选项集	支持度计数
{I1, I5, I9}	1
{I1, I6, I9}	1
{I1, I2, I10}	1
⋮	
{I2, I8, I10}	2
{I3, I8, I10}	1
⋮	

产生频繁集

频繁项集	支持度计数
{I2, I8, I10}	2

图 4-3　Apriori 算法实例流程

对于巨量候选项集的运算和开销并没有优化。下面介绍一个高效的基于哈希思想获取频繁集的 DHP 算法。

1. DHP 算法的核心思想

DHP 算法是一个典型的基于哈希思想的算法。哈希思想实质就是通过某种处理方式，尽量将要处理的所有数据放入内存，以提高处理效率。其实 Apriori 算法如果在计算

的过程中内存能够满足算法中数据处理的需求,是能够很好地执行的,但是当频繁项集的数量比较大以至于内存存放不了时,对这些项集的频繁计数将会使得内存中页面频繁地换入换出,这在操作系统中称为内存抖动,这种现象会大大降低执行效率。尤其是针对候选 2 项集 C_2,C_2 中包含的候选项集大小通常都非常大,因此许多优化算法针对这一问题提出了针对减少 C_2 大小的改进策略。通常在计算候选项集时特别是在计算候选项对时需要消耗大量内存,针对 C_2 候选项对过大的问题,一些算法被提出以减少 C_2 的大小,DHP 算法就是其中之一。

DHP 算法的核心思想是:在获取频繁项集的第一步里仅使用不到 10%的内存空间,因为只需要两个表,一个用来保存每个项的名字到一个整数的映射,随后的计算中用这些整数值代表项,另一个是定义一个数组来对这些整数代表的项进行计数,因此会有很多内存空间空闲。DHP 算法充分使用这些空闲的内存来另外定义了一个整数数组,将这个数组看作一个哈希表,表的每个桶中装的是代表各个项的整数值,在扫描计数获得每个项出现的次数时,项对同时也被散列到这些桶中,由此推出各个 2 项集是否为 2 频繁项集。归纳起来,DHP 算法主要有两个优化思想,即定义哈希表和用位图(bitmap)方式对其进行存储。

定义 4-13 生成哈希表。生成一个哈希表数组,当扫描数据集中的每个事务时,在由 C_1 候选集产生频繁项集 L_1 时,可以对每个事务同时产生所有的 2 项集,通过相应的散列函数将它们散列到表中不同的桶中,并随之增加对应的桶计数(图 4-5)。因此每个桶都对应一个数字,这个数字表达了桶中 2 项集的数目。显然如果桶中数字大于支持度阈值 s,这个桶被称为频繁桶。对于频繁桶,不能确定其包含的所有 2 项集是否为频繁的,但是有可能是的程度较高。相反,对于阈值低于 s 的桶,可以肯定其包含的 2 项集肯定不是频繁的,通过如此过滤可以大大减少候选 2 项集的数目。

例 4-6 候选 2 项集散列表的产生。

在计算 2 项集数据量比较大时,需要花费的资源就非常多。为提高性能引入了哈希表的概念,它的计算过程如下,首先计算哈希表长度,得到哈希桶大小,建立哈希桶。扫描数据库的每一个事务,找到每一个事务所包含的全部 2 项集,通过散列函数计算每一个候选 2 项集的桶地址。根据桶地址压入相应的哈希桶中,使相应的桶计数加 1,扫描整个数据库,把所有候选信息都压入哈希桶中,桶计数大于或等于最小支持度的候选 2 项集就是可能的频繁 2 项集,下面以实例说明。

如图 4-4 所示,频繁 1 项集 $L_1=\{I1,I2,I3,I4\}$,$|L_1|=4$,散列表长度 $|L_1|*(|L_1|-1)/2=6$,令 order(I1)=1,order(I2)=2,order(I3)=3,order(I4)=4。

扫描数据库里的每一个事务,事务{I1,I3,I5}中包含了{I1,I3},计算 $h(I1,I3)=2$,则{I1,I3}的哈希桶地址为 2,把地址为 2 的桶计数加 1。计算得到其他的候选 2 项集桶地址如图 4-5 所示。事务{I2,I4,I6}中包含了{I2,I4},计算 $h(I2,I4)=5$,则{I2,I4}的哈希地址为 5,把地址为 5 的桶计数加 1。同样事务{I1,I2,I3,I4}包含了{{I1,I2},{I1,I3},{I1,I4},{I2,I3},{I2,I4},{I3,I4}},则相应 2 项集的桶计数加 1。事务{I1,I2,I4}包含了{{I1,I2},{I1,I4},{I2,I4}},则相应 2 项集的桶计数加 1。事务{I1,I4,I8}包含了{I1,I4},则相应的桶计数加 1。最后通过桶计数和最小支持度比较,得到频繁 2 项集{{I1,I2},{I1,I3},{I1,I4},{I2,I4}}。

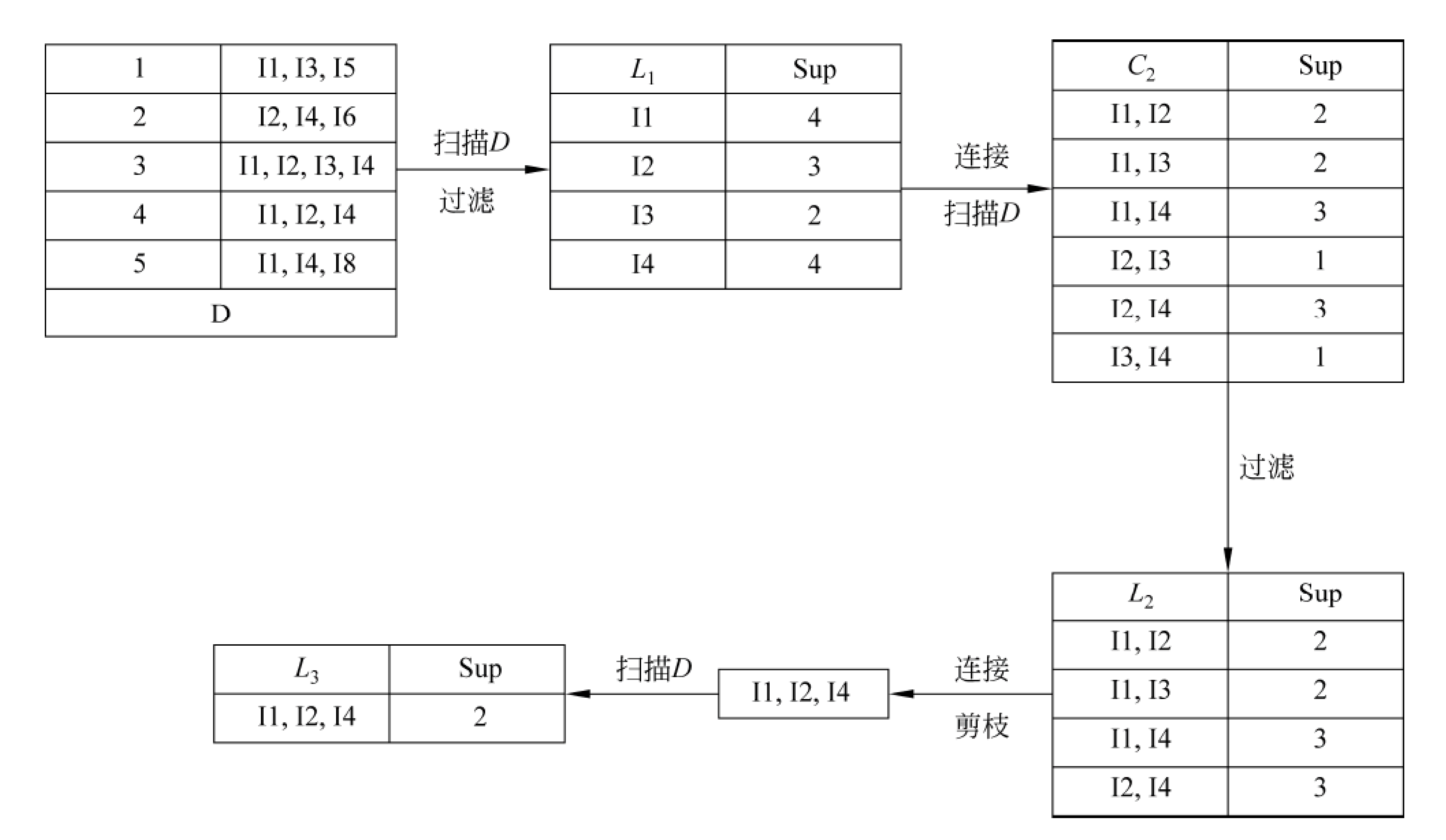


图 4-4　Apriori 算法流程

	I1, I2	I1, I3	I1, I4	I2, I3	I2, I4	I3, I4
桶计数	2	3	2	1	6	1
桶地址	1	2	3	4	5	6

图 4-5　生成的哈希桶示意图

定义 4-14　生成 bitmap 表。在 DHP 算法中，为了方便地获得哪些 2 项集是频繁的，哪些不是，通常哈希表应该常驻内存，如果数据集中包含的项太多，也就是大数据的情况下，哈希表也会过大。DHP 算法将哈希表压缩为 bitmap，位图中每一位代表一个桶。若桶为频繁桶，则位置为 1，否则置为 0。Bitmap 的使用就可以大大减少哈希表所占用的内存空间，从而保证 DHP 算法能直接在内存中较好地处理这个数据集而不会耗尽内存。

2. DHP 算法的伪代码描述

DHP 算法中使用哈希表的好处并不是绝对最优的，它还取决于数据大小和空闲内存大小。在最坏的情形下，所有的桶都是频繁的，那么 DHP 计算的项对与 Apriori 算法是一样的。然而，在通常实际使用中，大多数的桶都是非频繁的。这种情况下，DHP 算法能够大大缓解 Apriori 算法的瓶颈问题。下面详细给出该算法主要框架的伪代码，其中的子函数可以分别参考文献[1]和[9]。

算法 4-2　DHP 算法的伪代码描述

```
/ * Part 1 * /
s=最小支持度
设置 hash 表 H2 对应的所有 hash 桶均为 0      // 哈希表初始化
forall 事务 t∈D do begin
```

```
        对出现在哈希树中的1-项集进行插入和计数；
        forall 2-项集 x of t do
            H2[h2(x)]++；
    end
    L1={c|c.count>=s,c是哈希树的一个叶子节点}；
/* Part 2 */
    k=2；
    Dk=D；/* k项集的事务数据 */
    while(|{x|Hk[x]>=s}|>=LARGE){
        /* 创建哈希表 */
        gen_candidate(Lk-1，Hk,Ck)；
        设置 Hk+1所有的桶为0；
        Dk+1=ϕ；
    forall 事务 t∈Dk dobegin
    count_support(t,Ck,k,t1)；/* t1⊆t */
     if(|t1|>k)then do begin
     make_hash t(t1,Hk,k,Hk+1,t2)；
     if(|t2|>k)then Dk+1=Dk+1∪{t2}；
     end
     end
     Lk={c∈Ck|c.count>=s}；
     k++；
     }
/* Part3 */
    gen_candidate(Lk-1,Hk,Ck)；
    while(|Ck|>0){
    Dk+1=ϕ；
    forall 事务 t∈Dk dobegin
    count_support(t,Ck,k,t1)；/* t1⊆t */
    if(|t1|>k)then =Dk+1∪{t1}
    end
    Lk={c∈Ck|c.count>=s}；
    if(|Dk+1|=0)then break；
     Ck+1=apriori_gen(Lk)；
     k++；
     }
```

4.4 FP-Growth 算法

不同于上述两个算法都需要首先产生候选集，然后进行频繁项集验证的方式，FP-Growth 算法采用了模式段增长的方式分而治之地构建频繁项集。当数据集比较稠密

时，该算法使用扩展的前缀树 FP-Tree（Frequent-Pattern Tree），将全部的数据集压缩入主存，由此采用分治策略将对整体数据集的分析转化为对各个条件模式库分析的子任务，显著减小了搜索空间。由于 FP-Growth 算法降低了 I/O 开销，减少了同一时刻内存中存储的频繁项集数目，因而大大提高了算法运行性能。但是对于稀疏的大型数据集，该算法仍然不能保证一次装入内存。由于实际上 FP-Tree 对数据集的压缩率比较低，在这种情况下 FP-Growth 算法效率依旧会严重降低。

本节主要介绍关联规则的 FP-Growth 算法，针对比较稠密的数据集，该算法具有很好的计算效果，能够有效地解决大量 I/O 问题。在 4.4.1 节中详细地介绍 FP-Growth 算法的核心思想，该算法在 4.4.2 节中以伪代码表示形式进行了详细描述。

4.4.1 FP-Growth 算法的核心思想

首先给出该算法中的几个术语的定义。

定义 4-15 频繁模式树（frequent pattern tree）。简称为 FP-tree，它是指一棵将代表频繁项集的数据库压缩之后形成的树，该树仍保留项集的关联信息。其结构满足下列条件：由一个根节点（值为 null）、项前缀子树（作为子女）和一个频繁项头表组成。

定义 4-16 项前缀子树。该子树的每个节点包括 3 个域：item_name、count 和 node_link，其中，item_name 记录节点表示的项的标识；count 记录到达该节点的子路径的事务数；node_link 用于连接树中相同标识的下一个节点，如果不存在相同标识下一个节点，则值为 null。

定义 4-17 频繁项头表。本表中的每个表项包括一个频繁项的标识（item_name）和一个指向树中具有该频繁项标识的第一个频繁项节点的指针（head of node_link）。

定义 4-18 条件数据库（conditional data base）。又称为条件模式库，一种特殊类型的投影数据库，即把上述频繁模式树进行压缩后的数据库就是条件模式库。

定义 4-19 条件模式基（conditional pattern base）。在 FP-tree 中，每个任意的后缀模式可以形成各自的不同前缀子路径，所有这些路径组成了该后缀的条件模式基，即该后缀的子数据库。

定义 4-20 条件模式树（conditional pattern tree）。由某个后缀的条件模式基所构建的 FP-tree 称为该后缀的条件模式树。

基于 FP-tree 挖掘频繁项集，需要了解 FP-tree 的一些重要性质。

性质 4-1 节点链性质。对于任何频繁项 item，从 FP-tree 的项头表对应 item 项的头指针（head of node_link）开始，通过遍历 item 的节点链（node_link）可以挖掘出所有包含 item 的频繁模式。

性质 4-2 前缀路径性质。为了计算以 item 为后缀的频繁模式，仅仅需要在 FP-tree 中计算 item 节点的前缀路径，所有这些前缀路径的频繁度（计数）为该路径上该后缀 item 的频繁度（计数）。

性质 4-3 条件模式树构造性质。为了构造 item 的条件模式树，首先累加其每个条件模式基上所有其他 item 的频繁度（计数），过滤掉那些低于最小支持度阈值的其他 item，用剩下的 item 和该后缀一起构建 FP-tree。

4.4.2 FP-Growth 算法描述

FP-Growth 算法第一遍扫描数据集时，找出频繁 1 项集 L_1，对它们按降序排序；第二遍扫描数据集时，对每个事务过滤掉不频繁的项集，将剩下的频繁项集按 1 项集 L_1 中的顺序排序，把每个事务的频繁 1 项集插入到 FP-tree 中，相同前缀的路径可以共用，同时增加一个项头表，把 FP-tree 中的相同 item 连接起来，也是降序排序。在得到的 FP-tree 基础上进行频繁项集挖掘的过程如下：

(1) 从项头表的最下面的 item 开始，构造每个 item 的条件模式基。顺着项头表中每个 item 的链表，分别找出所有包含该 item 的前缀路径，这些前缀路径就是该 item 的条件模式基。所有这些条件模式基的频繁度(计数)满足性质 4-2。

(2) 通过采用性质 4-3 构造 FP-tree，递归地挖掘每个条件 FP-tree，累加后缀频繁项集，直到找到的 FP-tree 只有一条路径(或只有一个分支)，进一步通过组合的方式构建该分支中的所有频繁项集，或者当 FP-tree 为空时结束递归。

算法 4-3 FP-Growth 算法伪代码描述：

输入：

D：事务数据库

min_sup：最小支持度阈值

输出：频繁模式的完全集

Procedure1：/ * 构造 FP 树 * /

1. 一次扫描事务数据库 D，收集频繁项集 F 和它们的支持度计数，对 F 中的各项按支持度计数降序排列，结果构成频繁项列表 L

2. 创建 FP 树的根节点，以"null"标记。对于事务数据库 D 中的每一个事务，分别执行以下步骤：选择每个事务中的频繁项，并按照 L 列表中的次序排序。若经过排序后的频繁项列表为[p|P]，其中 p 为第一个元素，P 为剩余元素的列表，调用子函数 insert_tree([p|P]，T)，即创建 FP-tree。该函数的执行过程如下：若根节点 T 有子女 n 使得 n. item-name＝p. item-name，则相应子女节点 n 的计数加 1，否则创建一新节点 n，链接到根节点 T，并将其计数置为 1，同时将该节点链入项头表中具有相同 item-name 的节点上，递归调用 insert-tree，直至 P 为空。

Procedure 2：Procedure FP-growth(Tree，α)/ * 对于 FP-Growth 的调用如下：* /

```
if Tree 包含单个路径 P then
for 路径 P 中节点的每个组合(记作 β)
产生模式 β∪α，其支持度计数 sup_count 等于 β 中节点的最小支持度计数；
else for Tree 的项头表中每个 αi{
产生一个模式 β=αi∪α，其支持度计数 sup_count=αi * sup_count；
构造 β 的条件模式基，然后构造 β 的条件 FP 树 Treeβ；
if Treeβ≠ϕthen
调用 FP-Growth(Treeβ，β)；
}
```

例 4-7 FP-Growth 算法的实现过程。

数据如表 4-5 所示，假设最小支持度为 3。

表 4-5　购物篮抽取事务数据

购物篮 TID	商品 ID 列表	购物篮 TID	商品 ID 列表
T001	I6,I1,I3,I4,I7,I9,I13,I16	T004	I2,I3,I11,I18,I16
T002	I1,I2,I3,I6,I12,I13,I15	T005	I1,I6,I3,I5,I12,I16,I13,I14
T003	I2,I6,I8,I10,I15,I17		

根据以上数据，生成频繁 1 项集，并对其中各项目进行排序，得到 L_1＝{I6：4,I3：4,I1：3,I2：3,I13：3,I16：3}，压缩后的事务数据库如表 4-6 所示，生成项头表（如表 4-7 所示）和 FP-tree（如图 4-6 所示）。

表 4-6　压缩后的购物篮事务数据

购物篮 TID	商品 ID 列表	购物篮 TID	商品 ID 列表
T001	I6,I3,I1,I13,I16	T004	I3，I2,I16
T002	I6,I3,I1,I2,I13	T005	I6,I3，I1，I13，I16
T003	I6,I2		

表 4-7　事务数据项头表

项 ID	支持度计数	节点链	项 ID	支持度计数	节点链
I6	4	I6：4	I2	3	I2：3
I3	4	I3：3	I13	3	I13：2
I1	4	I1：3	I16	3	I16：2

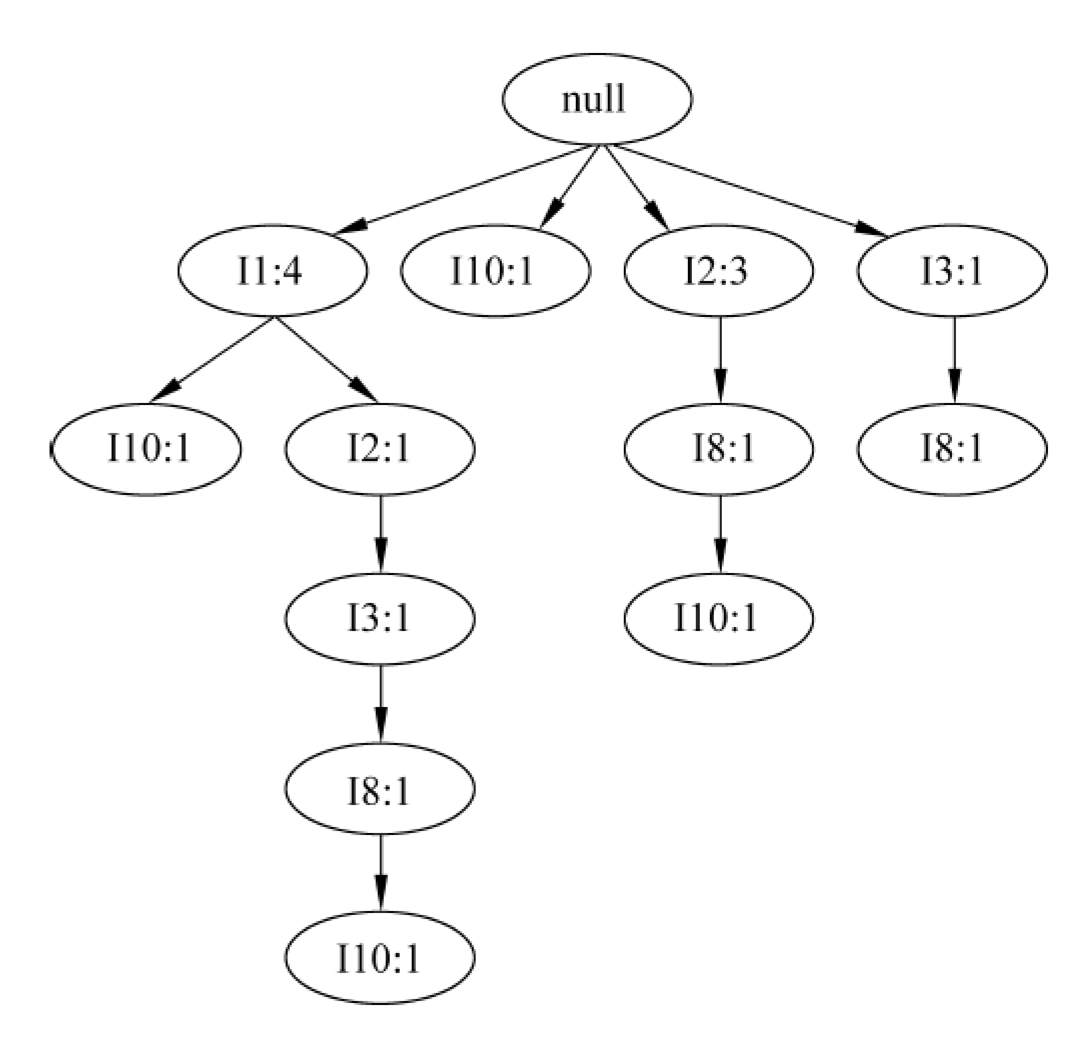


图 4-6　事务数据项头表对应的 FP-tree

FP 树的挖掘过程如下：由项头表的最后一项开始，对每一个长度为 1 的频繁集构造它的条件模式基（由 FP-tree 中与该后缀模式一起出现的前缀路径集组成），然后，构造它的条件 FP-tree，并递归地在该树上进行挖掘，模式增长通过后缀模式与条件 FP-tree 产生的频繁模式连接实现。得到条件模式如表 4-8 所示。

表 4-8 条件模式表

项 ID	条件模式基	条件 FP-tree	产生的频繁模式
I16	{{I6，I3，I1，I13：3}，{I3，I2：1}}	＜I3：3＞	{I16：4}，{I16，I3：4}
I13	{{I6，I3，I1：2}，{I6，I3，I1，I2：1}}	＜I6，I3，I1：3＞	{I13：3}，{I13，I6：3}，{I13，I3：3}，{I13，I1：3}，{I6，I13，I3：3}，{I1，I13，I3：3}，{I6，I13，I1：3}
I2	{{I6，I3，I1：1}，{I6：1}，{I3：1}}		{I2：3}
I1	{{I6，I3：3}}	＜I6，I3：3＞	{I1：3}，{I6，I1：3}，{I3，I1：3}，{I1，I3，I6：3}
I3	{I6：4}		{I6：4}

对于 I16，它的两个前缀形成条件模式基{{I6，I3，I1，I13：3}，{I3，I2：1}}，产生一个单节点的条件 FP 树＜I3：3＞，并导出一个频繁模式{I16，I3：3}。

同理，可得到由 FP-tree 产生的频繁模式，项头表如表 4-9 所示，FP-tree 如图 4-7 所示。

表 4-9 事务数据头表

项 ID	支持度计数	节点链	项 ID	支持度计数	节点链
I16	3	I16：3	I3	3	I3：3

针对 FP-Growth 算法的缺点，人们提出了 H-mine 算法、FP-Growth * 算法、CT-pro 算法、OP 算法等。它们主要在数据结构方面进行了优化，以便提高内存访问的效率。H-mine 算法从减小大数据集的内存消耗入手，构建了基于数组的 H-struct 结构，并采用分片思想，保证每一片都能够在使用时装入内存。FP-Growth * 算法使用局部性较好的数组存储 FP-Growth 算法中产生的条件模式库的支持度计数，在此基础上，构建条件模式树时，依靠该数组可以只扫描一遍原来的频繁模式树，明显提高了执行效率，该算法可以较好地运用在稀疏数据集上。CT-pro 算法使用了压缩的 FP-tree，其节点数是原来 FP-tree 树的一半，极大地节省了内存空间，提高了树的构建和遍历速度。OP 算法综合了 FP-Growth 算法和 H-mine 算法的优点，用树投影的方法处理稠密数据集，用数组结构处理稀疏数据集，而且能够根据数据集的动态特征动态地选取相关策略，在效率和空间上取得了较好的权衡。

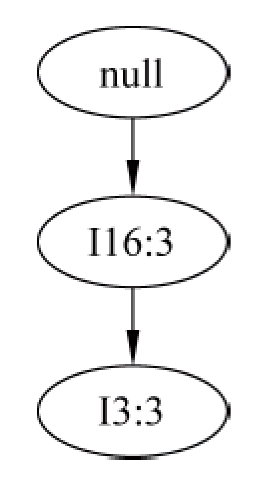


图 4-7 事务数据头表对应的 FP-tree

4.5 面向大数据的有效数据结构

在全球大数据浪潮中，对大数据进行智能分析和处理已经成为政府、企业等机构关注的焦点。数据分析方法所面对的数据集已经是海量数据，在算法中使用更优的数据结构，能够较好地利用内存，在节约系统开销的同时，提升算法的计算性能，接下来将讨论关联分析方法涉及的数据结构的特点以及相应的改进方式。

在许多关联分析算法中，主要的时间开销集中在将数据集中的事务从磁盘读入内存，进行频繁项集统计的这个过程上，因此当要处理的数据集文件太大以至于无法一次存放在内存中时，会出现存放数据的页面不断发生从内存到外存的换入换出操作，引起内存抖动现象，造成计算效率的急剧下降。另外当对数据集进行一遍扫描之后，相继获得的各个频繁项集都必须在内存中维护它们相对应的不同计数值，如果没有足够内存来存放这些数值，那么在之后的计算中随便对其中的一个加 1，都有可能发生一次页面的换入换出操作，内存抖动将非常严重，造成算法运行速度可能会比直接从内存中找到这些数值慢好几个数量级，所以应该尽量保证计算过程只发生在内存中已有的对象之间。

本节主要介绍大数据集在内存中的处理方法，针对大数据集在处理时的内存溢出或者内存抖动的问题，介绍了可供参考的几种有效数据结构。

1. 哈希结构

使用哈希结构，可以在生成候选集时过滤掉更多的项集，所以每一次生成的候选集都更加逼近频繁集，这种技术对于 2 项候选集的剪枝尤其有效。该技术首先扫描数据集，对所有的 1 项候选集进行计数，删除小于最小支持计数的 1 项集，与此同时，构建哈希表，在生成频繁 1 项集扫描的过程中，同时为每个事务产生所有 2 项集，并分别为这些 2 项集计算哈希地址存放到相应的散列桶中，每存放一次，该桶计数值加 1，一次数据集扫描结束后，2 项候选集中只保留那些对应散列桶计数值大于等于最小支持度阈值的 2 项集。同时为保证该哈希表能够一次读入内存，将其转换成位图，该位图的每一位对应一个散列桶，如果桶的计数值大于等于最小支持度阈值，则该位取值为 1，否则为 0。

2. 三角矩阵

任何对象要完成计算都必须首先调入内存，不在内存中的任何对象都不能进行计算，因此数据结构的定义应该尽量将更多的对象一次调入内存。由于关联分析中，往往 2 项集数目非常庞大，因此在将所有的频繁项集都编码成整数后，通过一遍扫描获取所有 2 项集的支持计数，进而可以将这个庞大的结果使用二维数组进行存储，显然这个二维数组是一个对称矩阵，对这类特殊矩阵的存储可以利用对称矩阵压缩存储的方法来存放计数结果，即使用一个一维的数组只存放这个矩阵中的上三角或下三角元素(不包含矩阵对角线上的元素)。这样对于一个 $n\times n$ 的二维矩阵，原来需要 n^2 个存储空间存放所有 2 项集的计数值，现在只需要 $n(n-1)/2$ 的存储空间。

3. 三元组

另一种可以用来减少2项集支持计数存放空间的方法是稀疏矩阵压缩存储中的三元组方法，即定义一个三元组，形如(i,j,c)，其中i和j分别对应于两个项目，c对应于扫描后获得的这两个项目同时出现在交易中的支持计数。由于在实际交易中往往会有大量的2项集支持计数值为0，所以对于这样的稀疏矩阵只要存储那些非0元即可，也就是可以采用三元组结构只存储支持计数非0的2项集，即只有$c>0$的才需要分配空间去存储。具体操作可以采用类似哈希表的数据结构，其中i,j是搜索键值，这样就能确定对于给定的i,j是否存在对应的三元组，如果是则快速定位。

如果所有可能出现的n^2个项对中至少有1/3共同出现在交易行为中，采用对称矩阵压缩存储方式更优。而如果出现的比例显著小于1/3，那么使用稀疏矩阵的三元组压缩存储方式更好。

4.6 关联规则有效性的评估方法

支持度和置信度是评价关联规则的两个常用的指标，通常一条通过数据分析得到的关联规则，只要它的支持度和置信度大于用户所设定的最小支持度和最小置信度，那么它就是强关联规则。但是是否所有的强关联规则都是可用的呢？试想一想，如果一条规则的后件本来的支持度就非常高，那么由此计算出来的该规则的置信度并不能说明这条规则的可用性。

例4-8 在一个交易数据库中共有10 000条交易记录，其中购买电子游戏的有6000人，购买VCD的人有7500人，而同时购买这二者的人数是4000人，假定最小支持度阈值为30%，最小置信度阈值为60%，对于一条如下的关联规则：

$$\text{电子游戏}\rightarrow \text{VCD}[\text{support}=40\%,\text{confidence}=66\%]$$

从表面上看，这条规则是一条满足条件的强关联规则，但是仔细分析会发现，该规则的后件VCD本身的支持度已经高达75%，必然会拉升规则的置信度值，使它超过最小置信度阈值。因此这并不能说明该规则的前件销售必然带来后件交易的增加，相反会发现，VCD原来的交易支持度已经达到了75%，而电子游戏的出现反而使其支持度只有66%，明显地说明了二者之间的一种负关系。

总之，判断一条强关联规则是否可用且有趣，可以采用两种评价方法，一种是客观评价方法，另一种是主观评价方法。客观评价方法应用统计学原理，用定量的数值来判定规则的有趣性，从而避免人为的主观臆断，可见客观性评价更可靠，更有说服力。但是客观评价完全基于数据而没有考虑到规则之间的联系和用户的感觉，所以目前在一些关联规则评价方法的研究中会增加一些主观度量方法或约束限制。比如，在强关联规则中，前件或者后件中部分项目可能对未来的决策和分析产生一定的影响，起到直接或间接的作用，这种作用表现在两个方面：①它们在整个规则中都表现出很重要的影响，常常引发需求者的关注；②具有一定的时效性，它们的出现和现实情况联系较为紧密。以上两种情况在确定关联规则的有趣性方面将会因需求而异。针对第一种

情况，相关领域的专家通常会给出相应的感兴趣度值，设置值域在 0 到 1 之间，0 为无关紧要的，1 为最重要的。针对第二种情况，会根据当时系统用户的感兴趣程度来赋值，因而表现出一定的时效性，值域同样为 0 到 1，最后评判该规则是否有用时，会通过衡量指标取二者的平均值来确定。

本节主要介绍关于关联规则有效性的几种评估方法。在 4.6.1 节中详细介绍了关联规则兴趣度的评估，进一步确认强关联规则的有效性。针对关联规则的最小支持度和置信度无法完全过滤无趣关联的问题，4.6.2 节中通过相关度的介绍对此类问题作了详细讲解。最后在 4.6.3 节讲解了关于关联规则度量的其他几类方法。

4.6.1 关联规则兴趣度评估

从上面的分析可以知道强规则不一定是有趣的，规则是否有趣可以主观或客观地评估。当规则发现之后，只有当时的用户能够评价这种规则是否有趣，并且这种判断是主观的，因人而异的。但是，根据交易数据"背后"的计数量，可以首先采用客观统计分析的兴趣度来过滤掉无趣的规则。

定义 4-21　正关联规则兴趣度(interest)。关联规则本身置信度与包含规则后件的交易支持度之间的差值，即规则的兴趣度＝规则的置信度－后件在事务数据库中的支持度。

也就是说，如果规则前件对后件没有任何影响，那么包含前件的交易中同时也包含后件的比例就应该等于所有交易中包含后件的比例，即该规则的兴趣度为 0。但是，不论是非正式还是严格意义上说，若一条规则的兴趣度很高或者是一个绝对值很大的负值，都是令用户十分关注的，前者说明在交易中前件的存在在某种程度上会促进后件的发生，而后者意味着前件的存在会抑制后件的发生。

例 4-9　在一个交易数据库中共有 10 000 条交易记录，其中购买电子游戏的有 6000 人，购买 VCD 的人有 7500 人，而同时购买这二者的人数是 4000 人，假定最小支持度阈值为 30%，最小置信度阈值为 60%，对于一条如下的关联规则：

电子游戏→VCD[support＝40%，confidence＝66%]

规则兴趣度(Interest)＝规则置信度－购买 VCD 的支持度＝－0.09

从上例可知，电子游戏的购买会抑制 VCD 的购买意愿。

定义 4-22　负关联规则兴趣度。对于一条负关联规则 $X \Rightarrow Y$，它的兴趣度计算公式定义如下：

$$RI = \frac{\text{exp_sup}(X) - \text{sup}(X \cup Y)}{\text{sup}(X)} \tag{4-4}$$

其中 exp_sup(X)是项集 X 的期望支持度。

4.6.2 关联规则相关度评估

正如前面所看到的，由于支持度和置信度不足以过滤掉那些无趣无用的关联规则，为了解决这个问题，除了可以用兴趣度来评估之外，通常也会定义相关度指标来扩充支持度和置信度的强关联规则判断框架。这就导致了描述一条真正强关联规则的方法，如下

所示：

$$A \rightarrow B[\text{support}, \text{confidence}, \text{correlation}] \tag{4-5}$$

以上公式不仅使用了支持度和置信度度量方式，而且加入了项集 A 和项集 B 的相关度指标。下面主要介绍一些常用的相关度评估指标。

1. 提升度度量

提升度(lift)是一种简单的相关度量，对于一条关联规则 $X \Rightarrow Y$，其提升度表示含有 X 的条件下同时含有 Y 的概率与 Y 总体发生的概率之比，其计算公式如下：

$$\text{lift}(X,Y) = \frac{P(Y \mid X)}{P(Y)} \tag{4-6}$$

如果上式的值小于1，则 X 的出现与 Y 的出现是负相关的，意味着一个出现可能导致另一个不出现；如果结果值大于1，则 X 和 Y 是正相关的，代表一个出现另一个也会出现；如果结果值等于1，则 X 和 Y 是独立的，它们之间没有相关性。

例 4-10 在一个交易数据库中共有 10 000 条交易记录，其中购买电子游戏的有 6000 人，购买 VCD 的人有 7500 人，而同时购买这二者的人数是 4000 人，假定最小支持度阈值为 30%，最小置信度阈值为 60%，这些事务可以汇总在一个相依表中，如表 4-10 所示。

表 4-10 汇总关于购买电子游戏和 VCD 事务 2×2 相依表

	电子游戏	$\overline{\text{电子游戏}}$	$\sum_{row}$
VCD	4000	3500	7500
$\overline{\text{VCD}}$	2000	500	2500
$\sum_{col}$	6000	4000	10 000

对于一条如下的关联规则：

$$\text{电子游戏} \rightarrow \text{VCD}[\text{support}=40\%, \text{confidence}=66\%]$$

$$\text{lift}(\text{电子游戏}, \text{VCD}) = \frac{P(\text{VCD} \mid \text{电子游戏})}{P(\text{VCD})} = 0.88$$

由计算结果可知，该强关联规则的提升度值为 0.88，说明规则的前件电子游戏的购买会抑制规则后件 VCD 的购买，将这条规则提供给用户是没有意义的。

2. 卡方度量

卡方(χ^2)度量的相关性计算在第 2 章中已经介绍过了，为了计算 χ^2 值，必须先得到规则前件和后件的相依表，举例如下。

例 4-11 在一个交易数据库中共有 10 000 条交易记录，其中购买电子游戏的有 6000 人，购买 VCD 的人有 7500 人，而同时购买这二者的人数是 4000 人，假定最小支持度阈值为 30%，最小置信度阈值为 60%，相依表上的观测值和期望值，如表 4-11 所示。对于一条如下的关联规则：

电子游戏 → VCD[support = 40%,confidence = 66%]

$$\chi^2 = \sum \frac{(观测值 - 期望值)^2}{期望值}$$

$$= \frac{(4000-4500)^2}{4500} + \frac{(3500-3000)^2}{3000} + \frac{(2000-1500)^2}{1500} + \frac{(500-1000)^2}{1000}$$

$$= 555.6$$

由于 χ^2 值大于 1,而且电子游戏和 VCD 的观测值 4000 小于期望值 4500,所以二者显然是负相关的。

表 4-11　显示期望值的相依表

	电子游戏	$\overline{电子游戏}$	$\sum_{row}$
VCD	4000(4500)	3500(3000)	7500
$\overline{VCD}$	2000(1500)	500(1000)	2500
$\sum_{col}$	6000	4000	10 000

4.6.3　其他相关评估度量方法

上面的讨论表明,简单地使用支持度-置信度框架来评估关联规则模式是不够的,还必须引入其他度量,如兴趣度、相关度等,这样才能真正揭示模式的内在联系和有趣性。然而仅仅这样度量,效果就一定很好吗?我们所分析的数据集本身还存在不同的特点,比如有些是倾斜的数据集,有些是不平衡数据集,有些数据集里对于我们要考察的项集所包含的零事务太多,这些都会影响以上讨论到的各种评价模式的指标,造成分析结果的可用性不高。近三十年来,有许多关于模式评估度量方法的研究,这里介绍几种公认为较好的可以度量以上特殊数据集的评估指标。首先介绍几个相关概念。

定义 4-23　不平衡比(Imbalance Ratio,IR)是指关联规则 $X \Rightarrow Y$ 的前件和后件所包含的项集 X 和 Y 在交易数据集中被包含的不平衡程度。其计算公式如下:

$$IR(X,Y) = \frac{|sup(X) - sup(Y)|}{sup(X) + sup(Y) - sup(X \cup Y)} \tag{4-7}$$

其中,分子是项集 X 和 Y 的支持度之差的绝对值,而分母是包含项集 X 或 Y 的事务数。如果 X 和 Y 在数据集中被包含的程度基本相同,该不平衡比之值为 0;否则,两者之差越大,不平衡比就越大。

定义 4-24　零事务(null-transaction)是指在所有的交易数据中不包含所考察的规则前件和后件项集的事务。从 4.6.2 节介绍的相关度评价指标提升度和 χ^2 的计算公式可以看出,计算概率时采用的都是整个数据库中事务的总数,因此必然受零事务影响非常大,不能很好地识别关联规则的有趣性。

定义 4-25　零不变性(null-invariant)是指规则的度量值独立于零事务的个数,即不受零事务影响的程度。零不变性是度量大型事务数据库中的关联规则的重要性质。

除了兴趣度和相关度指标,业内领域专家也提出了其他度量模式有效性的评估方法。这里主要介绍 4 种这样的度量:全置信度、最大置信度、余弦度量和 Kulczynski 度量。

定义 4-26 全置信度(All-confidence)的定义如下:

$$\text{all_conf}(X,Y)=\frac{\sup(X\cup Y)}{\max\{\sup(X),\sup(Y)\}}=\min\{P(X\mid Y),P(Y\mid X)\} \quad (4\text{-}8)$$

其中 $\max\{\sup(X),\sup(Y)\}$是 X 和 Y 的最大支持度。

定义 4-27 最大置信度(max_confidence)的定义如下:

$$\text{max_conf}(X,Y)=\max\{P(X\mid Y),P(Y\mid X)\} \quad (4\text{-}9)$$

定义 4-28 余弦度量的定义如下:

$$\begin{aligned}\text{cosin}(X,Y)&=\frac{P(X\cup Y)}{\sqrt{P(X)\times P(Y)}}=\frac{\sup(X\cup Y)}{\sqrt{\sup(X)\times\sup(Y)}}\\&=\sqrt{P(X\mid Y)\times P(Y\mid X)}\end{aligned} \quad (4\text{-}10)$$

定义 4-29 Kulczynski(Kulc)度量。该度量是波兰数学家 S. Kulczynski 于 1927 年提出的,它是两个置信度的平均值,更确切地说,它是两个条件概率的平均值:

$$\text{Kulc}(X,Y)=\frac{1}{2}(P(X\mid Y)+P(Y\mid X)) \quad (4\text{-}11)$$

例 4-12 在典型的数据集上比较 6 种模式评估度量。

电子游戏和 VCD 两种商品购买之间的关系可以通过把它们的购买历史记录汇总在表 4-12 的 2×2 相依表中来考察,其中像 gv 这样的表项表示包含电子游戏和 VCD 的多个事务。

表 4-12 两个项的 2×2 相依表

	电子游戏(g)	$\overline{电子游戏}(\bar{g})$	$\sum_{row}$
VCD(v)	gv	$\bar{g}v$	v
$\overline{VCD}(\bar{v})$	$g\bar{v}$	$\overline{gv}$	$\bar{v}$
$\sum_{col}$	g	$\bar{g}$	$\sum$

表 4-13 显示了一组事务数据集、它们对应的相依表项和 6 个评价度量的值。先考察前 4 个数据集 D_1 至 D_4。考察后面定义的 4 个相关度计算指标,从该表可以看出,g 和 v 在数据集 D_1 和 D_2 中是正关联的,在 D_3 中是负关联的,而在 D_4 中是中性的。对于 D_1 和 D_2,g 和 v 是正相关的,因为 gv(10000)显著大于 $\bar{g}v$(1000)和 $g\bar{v}$(1000)。直观地,对于购买电子游戏的人(g=10000+1000=11000)而言,他们非常可能也购买了 VCD(gv/g=10/11=91%),反之亦然。

表 4-13 使用不同数据集的相依表项比较和 6 种模式评估度量

数据集	gv	$\bar{g}v$	$g\bar{v}$	$\overline{gv}$	χ^2	提升度	全置信度	最大置信度	Kulc	余弦
D_1	10000	1000	1000	100000	90557	9.26	0.91	0.91	0.91	0.91
D_2	10000	1000	1000	100	0	1	0.91	0.91	0.91	0.91
D_3	100	1000	1000	100000	670	8.44	0.09	0.09	0.09	0.09

续表

数据集	gv	$\bar{g}v$	$g\bar{v}$	$\overline{gv}$	χ^2	提升度	全置信度	最大置信度	Kulc	余弦
D_4	1000	1000	1000	100000	24740	25.75	0.5	0.5	0.5	0.5
D_5	1000	100	10000	100000	8173	9.18	0.09	0.09	0.09	0.09
D_6	1000	10	100000	100000	965	1.97	0.01	0.99	0.50	0.10

后 4 种度量在前两个数据集上均产生了度量值 0.91，显示 g 和 v 是强正相关联的。然而，提升度和 χ^2 对前两个数据集计算出的相关度结果却与后 4 个指标不一致，D_1 是强相关的，D_2 却是独立的，原因在于计算中对$\overline{gv}$的敏感性，产生了显著不同的结果。事实上，在许多实际情况下，$\overline{gv}$通常都很大且不稳定。例如，在购物篮数据库中，事物的总数可能按天波动，并且显著超过包含任意特定商品集的事务数。因此，一个好的度量相关度的指标不应该受不包含感兴趣项的事务影响，否则会产生不可信的结果。

同理，在 D_3 数据集上，后 4 个度量都正确的表明 g 和 v 是强负相关联，因为 g 和 v 之比等于 gv 和 g 之比，即 100/1100＝9.1%。然而，提升度和 χ^2 都错误地与此相悖。

对于数据集 D_4，提升度和 χ^2 都显示了 g 和 v 之间正关联，而其他度量显示“中性”关联，因为 gv 与 $\bar{g}v$ 之比等于 gv 与 $g\bar{v}$ 之比，均等于 1。这意味着如果一位顾客购买了电子游戏，则也会购买 VCD 的概率恰好为 50%。

全置信度、最大置信度、余弦度量和 Kulc 度量 4 个相关度量中都具有很好的零不变性，原因在于这 4 个度量仅受 X、Y 和 $X \cup Y$ 的个数的影响，更准确地说，仅受条件概率 $P(X|Y)$和 $P(Y|X)$的影响，而不受事务总个数的影响。但当上面 4 个相关度量值在 0.5 附近时，不容易断定是正相关还是负相关，此时可以结合使用提升度或卡方分析。然而由于大型数据集常常具有许多零事务，因此在进行相关分析选择合适的兴趣度量指标评价规则的有效性时，考虑零不变性是非常重要的，这里所讨论的 4 个零不变性度量非常重要，但是当数据集对于考察的规则的项集具有不平衡性或倾斜时，通常使用 Kulc 度量比较好。

总之，如果只是使用支持度和置信度来挖掘关联，可能产生大量规则。但是其中有很多规则是无用的，是用户所不感兴趣的。因此必须使用附加的度量来分析有趣有用的关联规则，使规则数量得到很好的控制，同时促使有意义的规则呈现出来。

4.7 多维关联规则挖掘

到目前为止，我们研究的基本上都是同一个字段的值，也就是同一个维上的关系，比如用户购买的物品关联规则“电子游戏⇒VCD”，然而在用于数据分析的商品交易数据仓库中，实际存储的数据往往是多维的，除了在销售事务中记录有购买的商品之外，还会记录购买商品的顾客信息、销售商品的时间、地点及商品的型号等附加信息。如果沿用数据库中的概念术语，上面的那条关联规则只有一个维，即商品维，而实际上决定销售数量和好坏的不仅有商品本身，还应该包括销售商品的时间、地点及顾客群体特征。一个好的关

联规则必然和多个维相关,只有包含有多个维的关联规则才能够提供关于现实世界更为有用的信息。然而传统的关联规则分析算法都只限于单维关联规则的挖掘,因此对这方面的研究将是一件十分具有实际意义和广泛应用背景的工作。为了便于读者更好地理解多维关联规则,首先介绍一些相关的概念,沿用数据库的术语,规则中的维也可以称为谓词。

本节主要介绍在多维数据的情况下关联规则的挖掘方法。给出了相关概念的定义,同时给出了针对数值属性所采用的4种分析方法,以及对静态和动态属性离散化的处理方式。下面在各种规则两端出现的 X 均代表所分析的交易数据集。

定义 4-30 单维(single-dimensional)或维内关联规则(intradimensional association rule)。在一条关联规则的前件和后件中都只有同一个谓词出现。例如:

$$\text{buys}(X,\text{"电子游戏"}) \Rightarrow \text{buys}(X,\text{"VCD"}) \tag{4-12}$$

定义 4-31 多维关联规则(multidimensional association rule)。是指涉及两个或多个维或谓词的关联规则。例如

$$\text{age}(X,\text{"20...29"}) \wedge \text{occupation}(X,\text{"student"}) \Rightarrow \text{buys}(X,\text{"laptop"}) \tag{4-13}$$

定义 4-32 维间关联规则(interdimension association rule)。是指不允许规则中所涉及的维或谓词重复出现的关联规则,如式(4-13)就属于此类。

定义 4-33 混合维关联规则(hybrid-dimension association rule)。是指允许规则中所涉及的维或谓词在规则的左右两边同时出现的关联规则。

$$\text{age}(X,\text{"20...29"}) \wedge \text{buys}(X,\text{"ipad"}) \Rightarrow \text{buys}(X,\text{"iphone"}) \tag{4-14}$$

在挖掘维间关联规则和混合维关联规则的时候,还要考虑不同的字段属性,如标称属性和数值属性。对于标称属性,原先的算法都可以处理;而对于数值属性,需要进行一定的处理之后才可以进行。处理数值属性的方法基本上有以下几种:

(1) 数值字段被分成一些预定义的层次结构。这些层次结构都是由用户预先定义的,得出的规则也叫做静态数量关联规则。

(2) 数值字段根据数据的分布分成了一些布尔字段。每个布尔字段都表示一个数值字段的区间,落在其中则为1,反之为0,这种分法是动态的,得出的规则叫布尔数量关联规则。

(3) 数值字段被分成一些能体现其含义的区间。它考虑了数据之间的距离的因素,得出的规则叫基于距离的关联规则。

(4) 直接用数值字段中的原始数据进行分析。使用一些统计的方法对数值字段的值进行分析,并且结合多层关联规则的概念,在多个层次之间进行比较,从而得出一些有用的规则,得出的规则叫多层数量关联规则。

相比于混合维关联规则,维间关联规则的研究比较成熟,以下将简单介绍几种仅限于挖掘维间关联规则的常用方法。在维间关联规则的频繁集搜索中,与单维关联规则挖掘不同,它不是搜索频繁项集,而是搜索频繁谓词集,例如搜索 k 谓词集就是搜索频繁的 k 个合取谓词集。

1. 使用数值属性的静态离散化挖掘多维关联规则

使用数值属性的静态离散化挖掘多维关联规则即使用概念离散化的方法对数值属性进行离散化。这种离散化在挖掘之前进行，数值属性的值用区间替代。如果任务相关的结果数据存放在关系表中，则 Apriori 算法只需要稍加修改就可以找出所有的频繁谓词集，而不是频繁项集(即通过搜索所有的相关属性，而不是仅搜索一个属性)，找出所有的频繁 k 谓词集将需要 k 或 $k+1$ 次表扫描。还可以结合其他策略如散列、划分和选样以改进性能。

2. 使用数值属性的动态离散化挖掘量化关联规则

首先根据数据的分布，将数值属性动态地离散化到“箱”，这些箱可能在挖掘过程会被进一步组合，因此说这个离散化过程是动态的，组合目的是为了满足某种挖掘标准，如最大化所挖掘的规则的置信度。由于这种方法将数值属性的值处理成量，而不是区间标号之类，其挖掘出来的关联规则称为量化关联规则。典型的代表是 ARCS(Association Rule Clustering System，关联规则聚类系统)算法。该方法的挖掘思想源于图像处理，本质上就是将量化属性对映射到满足分类属性条件的 2D 栅格上，然后搜索栅格点进行聚类，由此产生关联规则。ARCS 算法的步骤如下：

(1) 分箱。数值属性可能具有很宽的值域。以 age 和 income 为例，每个 age 值对应在一个平面栅格的 x 轴上有一个唯一的位置，类似地，每个 income 的可能值在 y 轴上有一个唯一的位置，为了使得这个平面压缩到一个可管理的尺寸，将数值属性的坐标离散化到区间，这些区间可以根据挖掘期间的要求动态进行合并，其中的分箱策略可以采用等宽或等深的方法。将上述产生的两个数值属性的每种可能进行组合，得到一个 2D 数组。

(2) 查找频繁谓词集。一旦 2D 数组设置好，就可以扫描它，以找出满足最小支持度的频繁谓词集。

(3) 关联规则聚类。采用类似于前面介绍的关联规则生成算法产生关联规则，将得到的强关联规则映射到 2D 栅格上，然后对这些规则进行组合或聚类，形成一条汇总的规则，以取代零散的规则。图 4-8 显示了给定数值属性 age 和 income，预测规则后件 buys(X,"laptop")的 2D 量化关联规则，从图上可见，这些规则紧密相连，所以可以进行合并，得到规则 4-19：

$$\text{age}(X,34) \wedge \text{income}(X,\text{"31k...40k"}) \Rightarrow \text{buys}(X,\text{"laptop"}) \quad (4\text{-}15)$$

$$\text{age}(X,35) \wedge \text{income}(X,\text{"31k...40k"}) \Rightarrow \text{buys}(X,\text{"laptop"}) \quad (4\text{-}16)$$

$$\text{age}(X,34) \wedge \text{income}(X,\text{"41k...50k"}) \Rightarrow \text{buys}(X,\text{"laptop"}) \quad (4\text{-}17)$$

$$\text{age}(X,35) \wedge \text{income}(X,\text{"41k...50k"}) \Rightarrow \text{buys}(X,\text{"laptop"}) \quad (4\text{-}18)$$

$$\text{age}(X,\text{"34...35"}) \wedge \text{income}(X,\text{"31k...50k"}) \Rightarrow \text{buys}(X,\text{"laptop"}) \quad (4\text{-}19)$$

(4) 优化。依据用户满意的关联规则要求，对求取强关联规则的最小支持度和最小置信度值进行启发式优化，提升关联规则的质量。

3. 挖掘基于距离的关联规则

量化关联规则获取时，要将数值属性进行分箱的离散化，但是这种离散化是机械的，

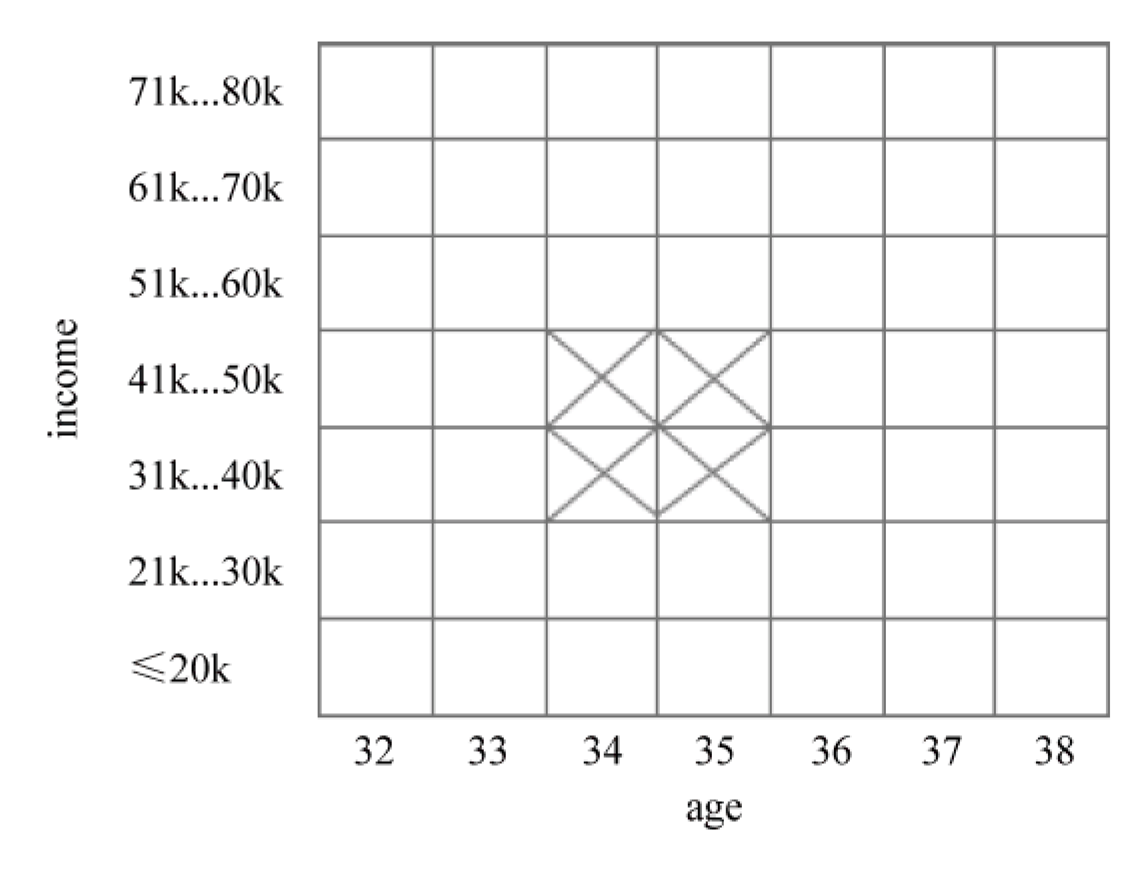


图 4-8 给定的数值属性 age 和 income

等宽分箱可能使距离很近的值分开，并且有可能创建一个没有值的区间，而等深分箱可能将很远的值放在一组，由此产生的离散化往往缺乏意义。基于距离的划分既能够考虑稠密性或区间内的点数，又可以考虑一个区间内点的“接近性”，因此可以产生更有意义的离散化。R. J. Miller 提出了一个非常经典的基于距离的算法，其主要分为两个步骤：

（1）使用聚类找出区间或簇。定义一个直径度量，评估数据对象的接近性，这个直径是投影到某个属性 X 上的数据对象两两距离的平均值，直径越小，其投影到属性 X 上越接近，因此直径度量可以评估簇的稠密性。满足稠密度阈值和频繁度阈值的数据对象形成一个簇。

（2）将簇进行组合，形成基于距离的关联规则。例如一个简单的形如 $C_X \Rightarrow C_Y$ 的基于距离的关联规则，意味着代表 X 属性的簇 C_X 中的数据对象在投影到属性 Y 的簇 C_y 上时，应该落在后者的区间内，或者接近该区间。二者的距离越小，意味着它们之间的关联程度越强。

4.8 多层关联规则挖掘

对于许多应用，在较高层发现的强关联规则可能是常识性知识。这时我们希望在更具细节的层次发现新颖的模式。但是，在低层或在原始层可能存在比较零碎的模式，并且其中一些也只不过是较高层模式的特例化。所以多层关联规则的挖掘更多地引起人们的关注，如例 4-12 所示。

本节主要介绍多层关联规则的挖掘方法，目的在于从更细的层次发现更加新颖的关联规则，并且详细地给出了相关的定义和概念。同时针对多层规则挖掘问题，介绍了使用一致性最小支持度、递减支持度、基于组和项等的挖掘方式。

例 4-13 假定购买事务的相关数据集如表 4-14 所示，它是某超市商店的部分销售交易数据。将表中所涉及的商品进行抽象分层，由低层概念抽象到高层更一般的概念，如图 4-9 所示。

表 4-14 购买事务数据记录

TID	购买商品
T1	光明常温奶，天友冷藏酸奶
T2	多鲜白面包，回头客全麦面包，RIO 鸡尾酒
T3	可口可乐汽水，雪花啤酒，天友冷藏酸奶
T4	回头客全麦面包，光明常温奶，绿源果汁
⋮	⋮

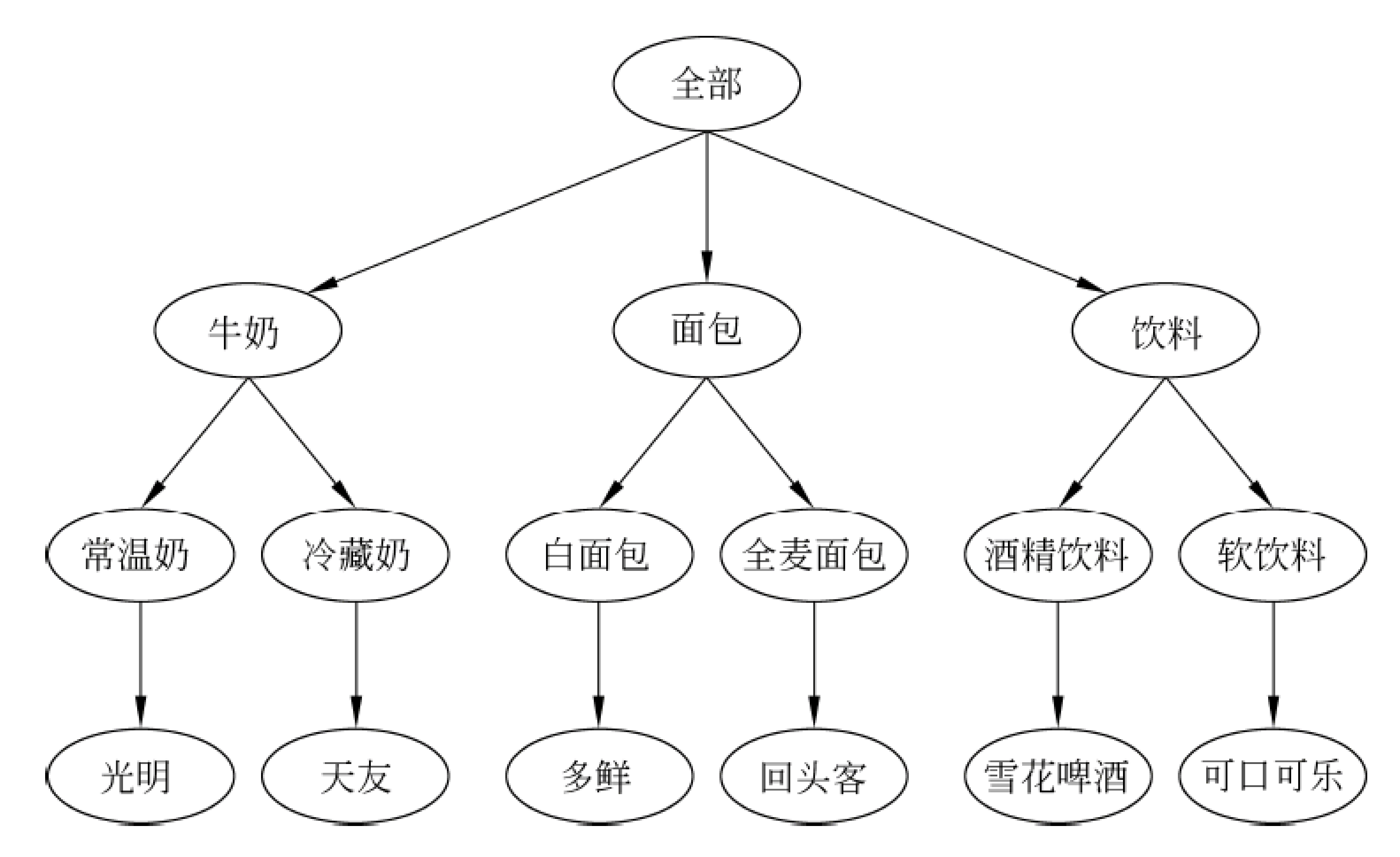


图 4-9 超市商品的概念分层

表 4-14 的项集都属于图 4-9 的最低层，通常在这样的原始数据层很难找到有趣的购买模式。例如光明常温奶和多鲜白面包，每个都是少量地出现在交易事务中，因此很难找到涉及它们的强关联规则，显然很少有人会同时购买这么具体的商品，因此“{光明常温奶，多鲜白面包}”不太可能满足最小支持度。但是把光明常温奶直接抽象化为常温奶，多鲜白面包抽象化为白面包更容易发生关联，且可能发生强关联。

作为传统单层关联规则挖掘技术的拓展，多层关联规则挖掘的研究方向通常分为同层关联规则挖掘和跨层关联规则挖掘两个方面。跨层关联规则最早是由 R. Srikant 和 R. Agrawal 在 1995 年提出的，他们在分析了具有分类特性数据的基础上，指出不同层也可能存在人们感兴趣的关联规则，并提出了一个多层关联规则挖掘算法，即 Cumulate 算法。此后诸多的研究人员和学者对多层关联规则的挖掘算法进行了大量的研究。目前，已提出的多层关联规则挖掘算法大多是通过对 Apriori 算法进行扩展得到的，例如 R. Srikant 和 R. Agrawal 提出的 Cumulate 算法、ML_T2 算法等。这类算法的核心仍源于 Apriori 算法，需要先求候选集，再求候选集的支持度。例如，ML_T2 算法采用自顶向下的策略，在每个层中采用 Apriori 算法挖掘每层的频繁模式，算法扫描的次数取决于频繁模式的长度。

如今的许多数据库或者数据仓库存储着大量的数据，在这样的数据中挖掘关联规则

需要很强的处理能力，分布式系统是一种解决方案，一些研究者提出了在分布式环境下对于每一层使用不同支持度的多层关联规则挖掘算法，该方法采用轮询方式处理分布式系统中各个节点间的通信问题，在各个节点上利用集合的“或”和“与”运算，在求候选频繁模式的同时就求出了模式的支持度，减少了数据库的扫描次数。随着空间和地理数据的大量累积，空间知识发现(SKD)和地理知识发现(GKD)逐渐成为相关研究领域的热点。研究人员以定性空间推理的 RCC 理论为基础，结合模糊逻辑，提出了一种面向空间数据库的近似区域空间关系模型，在此基础上给出了多层空间关联规则的挖掘算法 QSRSAR，该算法首先使用了 MBR(最小外包矩形)优先判定、顶点近似等手段针对大型空间数据库进行了优化预处理，进而在此基础上进行多层关联规则的挖掘。

定义 4-34 多层关联规则(multi-level association rules)。在多个抽象层的数据上挖掘产生的关联规则，这种挖掘通常采用自顶向下的策略，在置信度-支持度的框架下，从最高的概念抽象层开始，向下到较低的、更特定的概念层，在每个概念层累积计数，计算频繁项集，直到不能再找到频繁项集，对于每一层，可以使用发现频繁项集的任何算法。

在多层关联规则挖掘中，对每一抽象层频繁项集最小支持度设置非常重要。由于较低层次的项不大可能像较高层次的项出现得那么频繁，如果最小支持度设置得太高，可能丢掉出现在较低层次中有意义的关联规则；如果设置得太低，可能产生许多较高层无意义的关联规则。因此在不同的研究中，对每一层项集的最小支持度会有不同方式的设置，通常有以下几种。

(1) 对于所有层使用一致的最小支持度。

在每一个抽象层挖掘时使用相同的最小支持度阈值。如图 4-10 所示，对所有抽象层上频繁项集的挖掘都使用最小支持度阈值 5%。

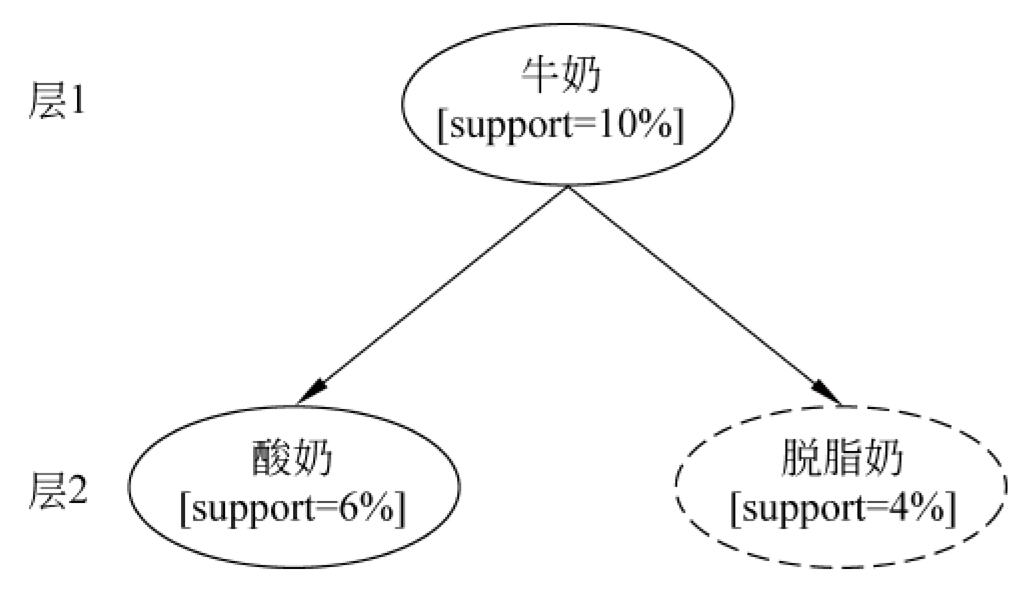


图 4-10 使用相同最小支持度阈值

使用一致的最小支持度阈值时，搜索的过程变得简单。用户只需要指定一个最小支持度阈值，根据祖先是其后代超集的相关知识，可以采用只搜索闭频繁集或者极大频繁集的优化策略进行频繁项集的查找，大大减少了查找时间。然而正如前面提到的一致支持度方法也有一些缺点一样，较低抽象层的项不可能像较高层的项出现得那么频繁。如果最小支持度阈值设置得太高，可能错失出现较低抽象层中有意义的关联规则；反之，则可能产生过多较高层中无意义的关联规则。

(2) 在较低层使用递减的最小支持度。

每一个抽象层拥有自己的最小支持度阈值。如图 4-11 所示，层 1 和层 2 的最小支持

度阈值分别是 5%和 3%。使用这种方法,会得到"牛奶""酸奶"和"脱脂奶"都是频繁项的结论。

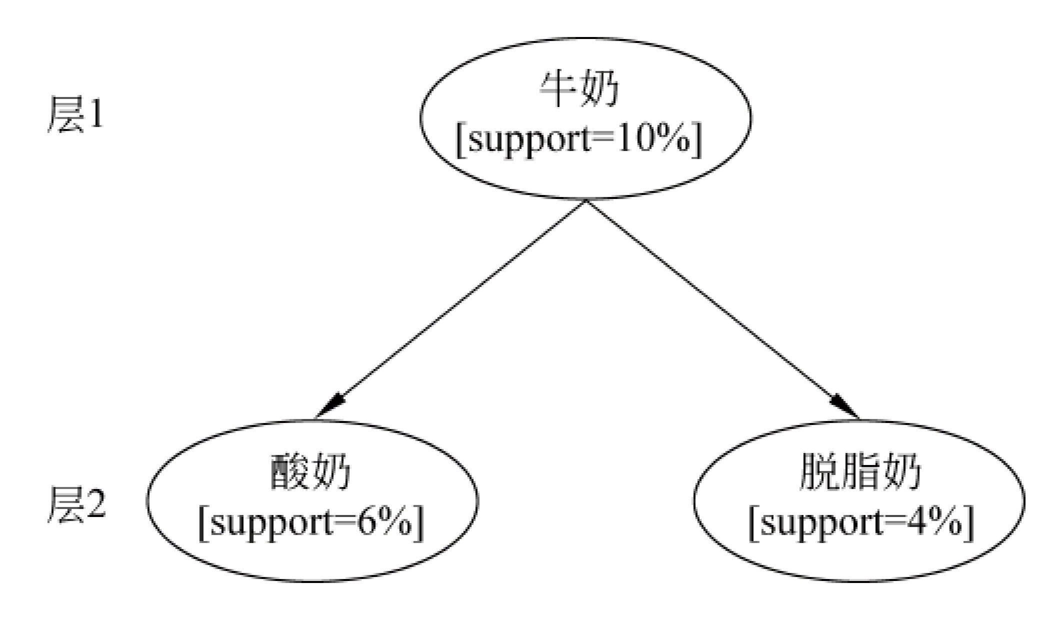


图 4-11 递减的最小支持度的多层挖掘

(3) 使用基于项或基于分组的最小支持度。

在挖掘多层关联规则时,有时更希望建立用户指定的基于项或基于分组的最小支持度阈值,因为用户和专家通常更清楚哪些组比其他组更重要。例如,超市经理更想了解"售价大于 20 元/盒的 950ml 纯鲜奶"的销售情况,为了能够发掘这类商品的关联模式,他们可以为此类商品的支持度设置更小的支持度阈值。

递减的最小支持度的多层关联规则挖掘能够搜索到更多有用的关联模式,许多研究和实际应用中采用了多种递减搜索的策略。

(1) 层交叉单项过滤。

一个第 i 层的项被考察,当且仅当它在第 $i-1$ 层的父节点是频繁的。如图 4-12 所示,由于父节点"牛奶"低于所在层的最小支持度,因此其子节点"酸奶"和"脱脂奶"不可能成为频繁集。

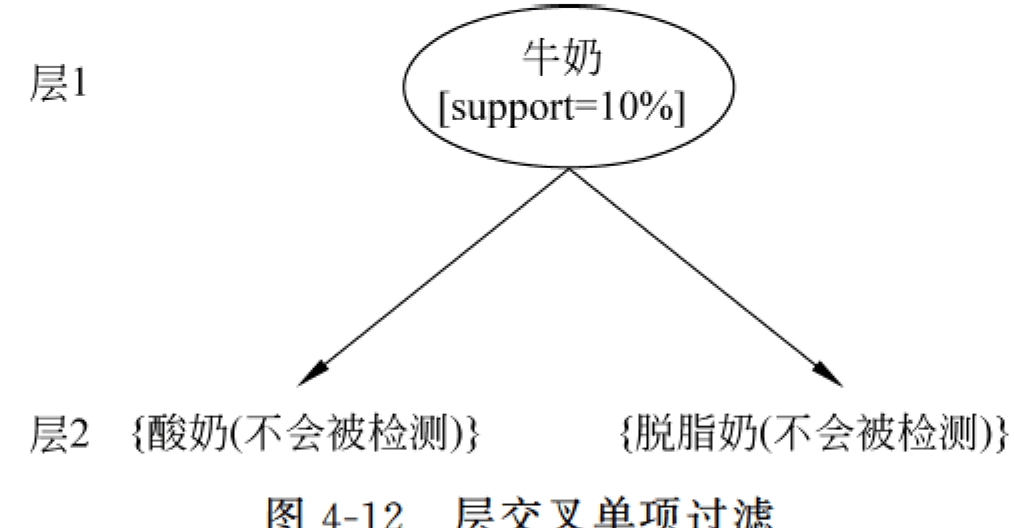


图 4-12 层交叉单项过滤

(2) 层交叉 k 项集过滤。

一个第 i 层的 k 项集被考察,当且仅当它在第 $i-1$ 层的对应父节点 k 项集是频繁的。如图 4-13 所示,由于父节点 2 项集"牛奶和面包"是频繁集,因此其子节点 2 项集"脱脂奶和白面包""酸奶和白面包""脱脂奶和黑面包"以及"酸奶和黑面包"都必须考察是否是频繁项集。

(3) 受控的层交叉单项过滤策略。

设置一个层传递阈值,用于向较低层传递相对频繁的项。如图 4-14 所示,虽然父节点"牛奶"的支持度小于本层的最小支持度阈值,但是所设置的层传递阈值为 8%,其支持

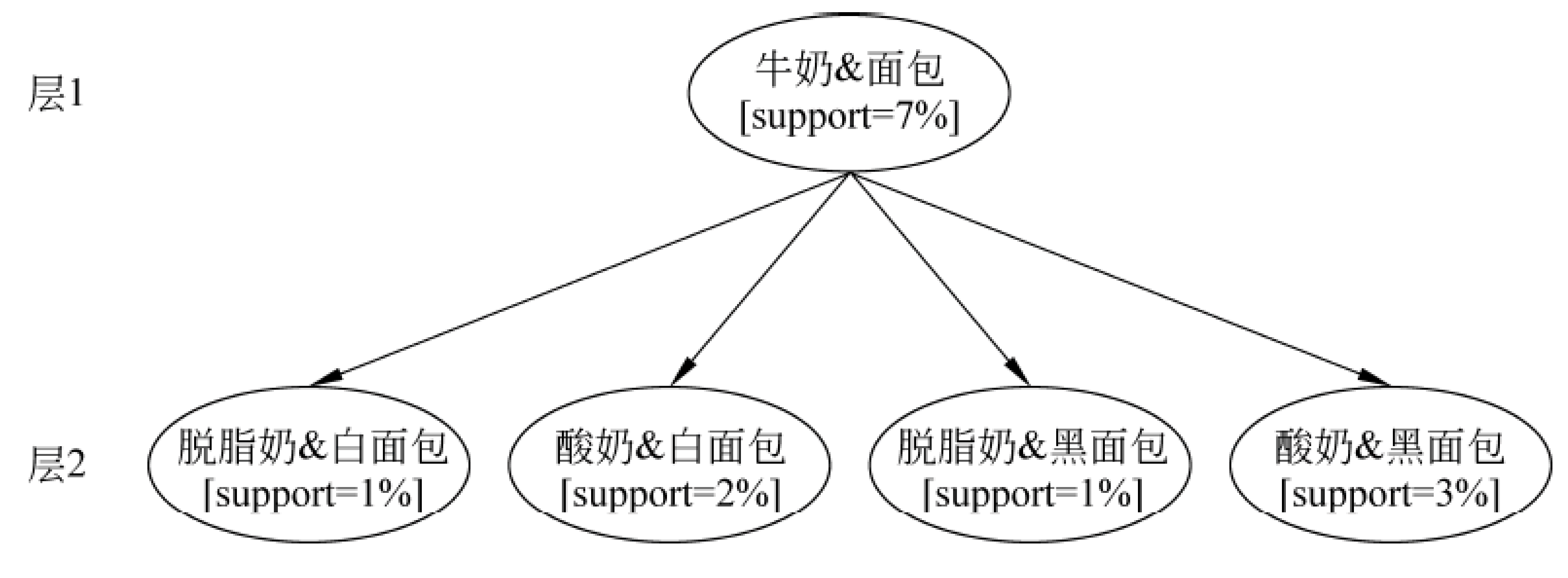


图 4-13 层交叉 k 项集过滤

度大于 8%，因此其子节点“酸奶”和“脱脂奶”都要被考察是否为频繁项集。

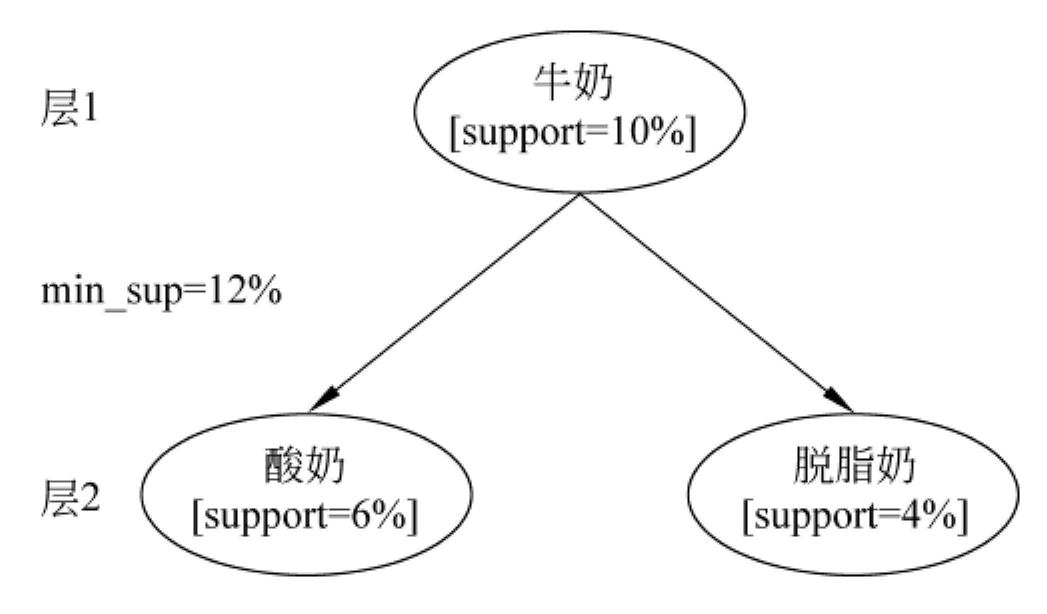


图 4-14 受控的层交叉单项过滤策略

(4) 交叉层关联规则。

应当使用较低层的最小支持度阈值，使得较低层的项可以包含在分析中，其中规则中的项不要求属于同一抽象层。

定义 4-35 多层关联规则的冗余性。在挖掘多层关联规则时，由于项间的“祖先”关系，有些发现的关联规则时常是冗余的，也就是说如果一个一般性的规则不提供新的信息，则是一个无趣和冗余的规则。通常根据此规则的祖先规则的支持度和置信度进行判断，如果它的支持度和置信度都接近于“期望值”，则被认为是冗余的。

例 4-14 多层关联规则的冗余性。

$$\begin{cases}\text{buys}(X,\text{"酸奶"}) \Rightarrow \text{buys}(X,\text{"白面包"}) \\ [\text{support} = 8\%, \text{confidence} = 70\%]\end{cases} \tag{4-20}$$

$$\begin{cases}\text{buys}(X,\text{"光明酸奶"}) \Rightarrow \text{buys}(X,\text{"沁园白面包"}) \\ [\text{support} = 2\%, \text{confidence} = 72\%]\end{cases} \tag{4-21}$$

如果在超市的销售中大约有四分之一的酸奶都是光明酸奶，则由以上两式可知，式(4-21)的支持度正好是式(4-20)的四分之一，而置信度相当，因此规则(4-21)不能提供任何更多用于营销策略的有用有效的信息，它不是有趣的，应该作为冗余规则从所得到的关联规则中删除。

综上，可见在多层概念下找到的关联比在原始数据下找到的单层关联规则更有趣，同时也更有用。

4.9 基于 Python 平台的案例分析

美国沃尔玛对它的一家分店做了关联分析，以便发现物品之间的关联关系，对物品摆放进行合理布局，以此提高商品的销售。合理而高效的关联分析不仅带来有效的数据支撑，更为商业决策者提供了理论与实际相结合的参考。

下面对沃尔玛实际的销售记录进行关联分析，借助于 Python 平台，发现商品之间的秘密。

1. 数据准备

本例提取了某沃尔玛分店一部分的销售记录，同时为了读者能够清晰地理解，这里过滤出了销售数目大于等于两件、小于七件的商品，并且属于食品的销售记录。总共获取记录 142 条，把这个数据文件命名为 Comm-rec-data。每一行代表一条销售记录，每条记录有以下几个特征：

- 交易流水号。
- 商品名称。

为了能够清晰理解商品销售记录的形式，下面选取了记录的前 20 条，如表 4-15 所示。

表 4-15 简易购物交易记录表

交易流水号	商品名称/商品编号					
0000	啤酒(1)			莴苣(4)	奶酪(3)	
0001		尿布(5)	啤酒(1)			火腿(6)
0002	奶酪(3)	尿布(5)	火腿(6)	啤酒(1)		
0003	火腿(6)	豆奶(2)			尿布(5)	
0004		啤酒(1)		莴苣(4)		奶酪(3)
0005			尿布(5)	火腿(6)		啤酒(1)
0006	啤酒(1)	奶酪(3)		莴苣(4)		
0007	尿布(5)		啤酒(1)		火腿(6)	
0008	奶酪(3)	豆奶(2)	啤酒(1)		尿布(5)	火腿(6)
0009			尿布(5)	豆奶(2)		
0010		啤酒(1)		莴苣(4)	奶酪(3)	
0011	豆奶(2)	啤酒(1)		尿布(5)		火腿(6)
0012			啤酒(1)		尿布(5)	奶酪(3)
0013		豆奶(2)			尿布(5)	火腿(6)
0014	奶酪(3)	莴苣(4)	啤酒(1)			

续表

交易流水号	商品名称/商品编号					
0015	尿布(5)	豆奶(2)		啤酒(1)		火腿(6)
0016			奶酪(3)	尿布(5)	啤酒(1)	
0017	豆奶(2)	尿布(5)	啤酒(1)			火腿(6)
0018	啤酒(1)		尿布(5)	莴苣(4)		
0019		奶酪(3)	啤酒(1)	莴苣(4)	尿布(5)	火腿(6)

分析了购物记录的基本特征后，为了能够让计算机识别其中的商品内容，程序设计时需要将记录中出现的各类商品进行编号，例如：0000 记录中的“啤酒”→1，“莴苣”→4 等，其他所有商品的编号见表 4-14 中每个商品名称后括号内的数字。

2. 规则获得的过程

完成以上的过程后，通过 Python 设计的关联规则界面对数据进行加载，获得频繁项集和关联规则，下面详细讲解整个过程。

根据频繁项集的概念，可得到{尿布，啤酒}为一个频繁项集。由此，再根据规则可以得到“尿布→啤酒”的关联规则，意味着买尿布的顾客很可能会买啤酒。使用频繁项集和关联规则能够帮助商家找到顾客的消费行为，商家能够及时地对这些行为进行一定的营销策略调整。

根据表 4-15 的数据，获取关联规则所需要的参数设定如下：

(1) **支持度**。一个项集的支持度(sup)被定义为数据集中包含该项集的数量占总计的比率。在表 4-15 所示的交易记录中，1 项集{尿布}的支持度是 0.40，2 项集{尿布，啤酒}的支持度是 0.30。

(2) **置信度**。又名可信度，对尿布和啤酒的关联规则定义可得到置信度为“项集{尿布，啤酒}的支持度/项集{尿布}的支持度”，所以根据公式可得到“尿布→啤酒”的置信度为 75%。

同理能得到啤酒和奶酪的关系，这里它们的置信度约为 70%，使用 Python 语言编写了 Apriori 算法，通过单击 loadData 加载数据文件，设置最小支持度为 0.5，最小可信度为 0.7，其他属性选用默认值，单击“开始”按钮，获得了如图 4-15 所示的关联规则。



图 4-15 Apriori 运行界面

所获得的频繁项集如图 4-16 所示。



图 4-16 Apriori 频繁项集

在图 4-16 所示的频繁项集中，获得了{1,3}的频繁项集，也就是{啤酒，奶酪}是频繁 2 项集，同时也能在关联规则中获得两者之间的规则。本 Apriori 算法用 Python 实现，在本书配套电子资源中提供了相关代码。

4.10 小 结

- 海量数据中频繁模式、相关关系和关联的发现在商品促销、商业决策和市场规划中是有用的。在众多此类应用中，**购物篮分析**是一种最热门的应用，它通过在购物事务记录中搜索常一起购买的商品集合，研究顾客的购买行为和习惯。
- 关联规则的挖掘过程主要是获得频繁项集，如在项集合中有“牛奶”和“面包”，它们满足**最小支持度阈值**或任务相关元组的百分比，由它们产生形如“牛奶⇒面包”的强关联规则。这些规则还满足**最小置信度阈值**(预定义的、在满足“购买牛奶”的条件下满足“购买面包”的概率)。可以进一步分析关联，发现项集“牛奶”和“面包”之间具有相关规则。
- 在频繁项集挖掘中，算法主要有 3 类：**Apriori 算法、改进的 DHP 算法和 FP-Growth 算法**。
- **Apriori 算法**是一种挖掘关联规则的频繁项集算法，其核心思想是通过候选集生成和向下封闭检测两个阶段来挖掘频繁项集。它逐层进行挖掘，利用了先验性质：频繁项集的所有非空子集也是频繁的。Apriori 算法的两个输入参数分别是最小支持度和数据集。该算法首先会生成所有单个元素的项集列表。接着扫描数据集来查看哪些项集满足最小支持度要求，那些不满足最小支持度的集合会被去掉。然后，对剩下来的集合进行组合以生成包含两个元素的项集。接下来，再重新扫描交易记录，去掉不满足最小支持度的项集。该过程重复进行，直到所有有项集被去掉。
- **频繁模式增长**(FP-Growth)算法基于 Apriori 构建，采用了高级的数据结构减少扫描次数，大大加快了算法速度。FP-Growth 算法只需要对数据库进行两次扫描，而 Apriori 算法对于每个潜在的频繁项集都会扫描数据集以判定给定模式是否频繁，因此 FP-Growth 算法的速度要比 Apriori 算法快。

- 并非所有的强关联规则都是有趣的。所以,应用适当的模式评估度量来扩展支持度-置信度框架,促进更加有趣的规则挖掘,以产生更加可行的相关规则。本章对**全置信度**、χ^2、**最大置信度**、**提升度**、**余弦度量**和 **Kulczynski 度量** 6 种评估度量进行了讨论,并说明了置信度、最大置信度、余弦度量和 Kulczynski 度量 4 种是零不变的。一般 Kulczynski 度量与不平衡比一起使用,提供了项集间的模式联系。

4.11 习　　题

1. 请证明下列关系:

$\text{sup}(X \cup Y \cup Z) \geq \text{sup}(X \cup Y) + \text{sup}(X \cup Z) - \text{sup}(X)$,此处 X、Y、Z 都是数据集里的频繁项集。

2. 试说明在 Apriori 算法中支持度和置信度扮演的角色。

3. 某大型量贩店为了了解顾客的消费行为以及产品组合的销售情形,定期搜集各收款机的每笔交易记录,以获得各时段中各类商品的购买次数及单笔数量。表 4-16 为抽样 5 笔交易记录(顾客所购买的商品组合),试回答下列问题:

(1) 利用 Apriori 算法找出所有 2 项集的可能规则。

(2) 计算所有 2 项集可能的规则支持度和置信度。

(3) 若支持度阈值定为 20%,且置信度阈值为 20%,试找出被列入候选集的规则。

(4) 利用 Apriori 算法找出所有的 3 项集的可能规则,计算其支持度与置信度,并找出被列入候选集的规则。

表 4-16　购物交易记录表

交易记录	商品名称和代码
160122330	面包(A)、甜酱(B)、芝麻酱(C)
160122331	面包(A)、芝麻酱(C)
160122332	面包(A)、芝麻酱(C)、牛奶(D)
160122333	面包(A)、啤酒(E)
160122334	牛奶(D)、啤酒(E)

4. 项集 X 称为数据集 D 上的生成元(generator),如果不存在真子集 $Y \subset X$ 使得 $\text{sup}(X) = \text{sup}(Y)$。生成元是频繁的生成元,如果 $\text{sup}(X)$ 满足最小支持度阈值。设 G 是数据集 D 上的所有频繁生成元的集合。

(1) 仅使用 G 和它们的支持度计数,你能确定项集 A 是否是频繁的,并且如果 A 是频繁的,确定 A 的支持度。如果可能,给出你的算法。如果不能给出算法,还需要什么信息?确定需要的信息,给出你的算法。

(2) 简述闭项集和生成元的关系。

5. 给出一个例子表明关联规则中的负模式。

6. 数据库有 5 个事务,如表 4-17 所示。设 min_sup=60%,min_conf=80%。

表 4-17　数据库事务记录表

TID	购买的商品	TID	购买的商品
T100	{M,O,N,K,E,Y}	T400	{M,U,C,K,Y}
T200	{D,O,N,K,E,Y}	T500	{C,O,O,K,I,E}
T300	{M,A,K,E}		

(1) 分别使用 Apriori 算法和 FP-Growth 算法找出频繁项集。比较两种挖掘过程的有效性。

(2) 列举所有与下面的元规则匹配的强关联规则(给出支持度 s 和置信度 c),其中,X 是代表顾客的变量,item 是表示项的变量(如"A""B"等):

$$\forall_x \in \text{transaction}, \text{buys}(X, \text{item}_1) \wedge \text{buys}(X, \text{item}_2) \Rightarrow \text{buys}(X, \text{item}_3)[s, c]$$

7. 请根据表 4-18 画出一棵 FP-tree 并解释如何从 FP-tree 中导出频繁项集。

表 4-18　数据库事务记录表

TID	项集	TID	项集
T01	{C,M,E}	T06	{A,C,E,O}
T02	{A,C}	T07	{B,C,O}
T03	{A,B,C,O,G}	T08	{C,E,G}
T04	{B,E,O}	T09	{B,C,E,G}
T05	{C,M,G}	T10	{B,C,G}

8. 大部分频繁模式挖掘算法只考虑事务中的不同项。然而,一种商品在一个购物篮中多次出现(如 4 盒牛奶或 3 份面包)的情况在销售数据分析中可能是重要的。考虑项的多次出现,如何有效地挖掘频繁项集?对著名的算法,如 Apriori 算法和 FP-Growth 算法,提出修改方案,以适应这类情况。

9. 假设作为一家连锁店经理,你想使用销售事务数据库评估你的商品广告效应,尤其是想研究具体因素如何影响预告特定类型的商品降价出售的广告效应。要研究的因素是顾客居住的地区(region)、星期几(day)和一天内的广告次数(times)。讨论如何设计一种有效的方法挖掘该事物数据集,并解释如何用多维和多层挖掘方法帮助你得到好的解。

10. 给定一个高频项目集{A,B,C,D,E,F},则在此项目集之下,最多可能存在多少条关联规则?

11. 假设一家生鲜超市的管理者想在固定时段将某些隔夜即需丢弃的商品以商品合售打折的方式进行推销,应该如何以关联分析来协助策划此方案?

12. 频繁模式挖掘可能产生过多的模式。所以,重要的是开发挖掘压缩模式的方法。假设用户只想得到 k 个模式(k 是一个小整数)。设计一种有效的方法,它产生 k 个最具代表性的模式,其中越是截然不同的模式越是首选的模式。使用一个数据量较小的数据集解释你的方法的有效性。

4.12 参考文献

[1] Agrawal R，Srikant S. Fast Algorithms for Mining Association Rules in Large Databases[J]. Journal of Computer Science and Technology，2000，15(6)：619-624.

[2] Liu Y X. Study on Application of Apriori Algorithm in Data Mining[C]. International Conference on Computer Modeling and Simulation，2010,3：111-114.

[3] Cheng Y，Xiong Y. Research and Improvement of Apriori Algorithm forAssociation Rules[C]. International Workshop on Intelligent Systems & Applications，2010：1-4

[4] 雷雨，李曼，胡卫松，等. 高效的稀有序列模式挖掘方法[J]. 计算机科学与探索，2015，9(4)：429-437.

[5] 陈才扣. 数据挖掘中负关联规则的研究[D]. 南京：东南大学，2000.

[6] Purdom，W. Paul，V. Gucht，et al. Average-case Performance of the Apriori Algorithm. SIAM J. Comput，2004，33(5)：1223-1260.

[7] Niu Z D，Nie Y X，zhou Q，et al. A Brain-Region-based Meta-Analysis Method Utilizing the Apriori Algorithm[J]. BMC Neuroscience，2016，17(1)：1-7.

[8] Wang F，Li Y H. An Improved Apriori Algorithm based on the Matrix[J]. International Seminar on Future Biomedical Information engineer，2008，30(10)：152-155.

[9] Park J S，Chen M S，Yu P S. An Effective Hash-based Algorithm for Mining Association Rules[C]. Proceedings of ACM SIGMOD International Conference on Management of Data，1995：175-186.

[10] Yang G，Zhao H，Wang L，et al. An Implementation of Improved Apriori Algorithm[C]. International Conference on Machine Learning & Cybernetics，2009，3：1565-1569.

[11] Han J W，Kamber M. 数据挖掘概念与技术[M]. 3 版. 范明，孟晓峰，等译. 北京：机械工业出版社，2012.

[12] Borgelt C. An Implementation of the FP-Growth Algorithm[C]. OSDM Proceedings of International Workshop on Open Source Data Mining Frequent Pattern，2010：1-5.

[13] 窦祥国. 关联规则评价方法研究[D]. 合肥：合肥工业大学，2005.

[14] Wang L，Fan X J，Long X L，et al. Mining Data Association based on a Revised FP-Growth Algorithm[C]. International Conference on Machine Learning and Cybernetics，2010：91-95.

[15] Deng Y G，She Y，Jia W Q. Research on the Improvement of FP-Growth based on Hash[C]. International Conference on Information Science &Engineering，2010：5362-5365.

[16] Zhu T Y，Ma Z X，Liu S J，et al. Improved Negative Selection Algorithm based on Bloom Filter [C]. International Conference on E-business & Information System Security，2009：1-4.

[17] 喻昌祺. 多维关联规则算法设计[D]. 北京：北京邮电大学，2006.

[18] Shanal J，Venkatachalam T. An Improved Method for Counting Frequent Itemsets Using Bloom Filter[C]. Procedia Computer Science，2015，47：84-91.

[19] Pontarelli S，Reviriego P，Maestro J A. Improving Counting Bloom Filter Performance with Fingerprints[J]. Information Processing Letters，2016，116(4)：304-309.

[20] Lent B，Swami A，Widom J. Clustering Association Rules[C]. International Conference on Data Engineering，1997：220-231.

[21] Padual R D，Santos F F D，Conrado M D S，et al. Subjective Evaluation of Labeling Methods for

Association Rule Clustering[C]. Mexican International Conference on Artificial Intelligence,2013, 8266: 289-300.

[22] Balcazar J L, Dogbey F. Evaluation of Association Rule Quality Measures through Feature Extraction[C]. Lecture Notes in Computer Science. 2013: 1-4

[23] Shimada K, Hanioka T. An Evolutionary Method for Exceptional Association Rule Set Discovery from Incomplete Database[J]. Information Technology in Bio-& Medical Informatic, 2014, 8649 (3): 133-147.

[24] Sawanta V, Shah K. Performance Evaluation of Distributed Association Rule Mining Algorithms [J]. Procedia Computer Science,2016, 79: 127-134.

[25] Miller R J, Yang Y. Association Rules over Interval Data[J]. ACM Sigmod Record, 1998, 26(2): 452-461.

[26] 刘大有,王生生,虞强源,等. 基于定性空间推理的多层空间关联规则挖掘算法[J]. 计算机研究与发展, 2004, 41(4): 565-570.

[27] 任家东,任东英,高伟. 分布式多层关联规则挖掘[J]. 计算机工程, 2003, 29(5): 96-98.

[28] Termier A, Rousset M C, Sebag M. A New Approach for Discovering Closed Frequent Trees in Heterogeneous Tree Databases[C]. IEEE International Conference on Data Mining. 2004: 543-546.

[29] Witten I H, Frank E, Hall M A. Data Mining: Practical Machine Learning Tools and Techniques [M]. 3rd ed,北京: 机械工业出版社, 2011.



数据分类分析

分类(classification)是一种重要的数据分析形式,它是提取刻画重要数据类的模型,也是机器学习和数据挖掘领域中的一整套用于处理分类问题的方法。该类方法是有监督学习类型的,即:给定一个数据集,所有实例都由一组属性来描述,每个实例仅属于一个类别,在给定数据集上运行可以学习得到一个从属性值到类别的映射,进而可使用该映射对新的未知实例进行分类,这种映射又称为模型或分类器(classifer)。在数据挖掘社区遴选出的十大算法中六个都是这类方法,这也反映出此类方法在数据挖掘中被使用的广泛程度。最早这类算法只能处理标称类别数据,如今已扩展到支持数值、符号乃至混合型的数据类型。具体的应用领域也很广泛,例如临床决策、生产制造、文档分析、生物信息学、空间数据建模(地理信息系统)等。

本章从介绍分类的基本概念(5.1 节)开始,其后,将学习数据分类分析的基本技术,包括最常用的决策树分类器的构建方法(5.2 节),基于概率统计思想的贝叶斯分类算法(5.3 节),具有统计学习理论坚实基础的,在所有知名的数据挖掘算法中最健壮、最准确的支持向量机(Support Vector Machine)算法(5.4 节),以及通过构建一组基于学习器进行集成学习的 Adaboost 算法(5.7 节),最后通过使用 Python 语言给出了一个具体案例,使读者能够熟悉数据分类分析的整个过程。

5.1 基本概念和术语

本节给出分类分析相关的基本概念及其基本术语,为读者研究分类分析建立基础。5.1.1 节通过一个描述性模型介绍分类中的有关定义。5.1.2 节介绍分类的方法,并对相关的术语做出了解释。

5.1.1 数据分类

分类任务就是通过学习得到一个目标函数(target function) f,把每个属性集 x 映射到一个预先定义的类标号 y。

目标函数也称分类模型(classfication model)。分类模型可以用于以下目的。

(1) 描述性建模。分类模型可以作为解释性工具,用于区分不同类中的对象。例如,对于生物学家或者其他人,一个描述性模型有助于概括表 5-1 中的数据,并说明哪些特征决定一种脊椎动物是哺乳类、爬行类、鸟类、鱼类或者两栖类。

表 5-1 脊椎动物的数据集

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
鲑鱼	冷血	鳞片	否	是	否	否	否	鱼类
鲸	恒温	毛发	是	是	是	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
鸽子	恒温	羽毛	否	是	是	是	否	鸟类
猫	恒温	软毛	是	否	否	是	否	哺乳类
豹纹鲨	冷血	鳞片	是	是	否	否	否	鱼类
海龟	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
鳗	冷血	鳞片	否	是	否	否	否	鸟类
蝾螈	冷血	无	否	半	否	是	是	两栖类

(2) 预测性建模。分类模型还可以用于预测未知记录的类标号。如图 5-1 所示，分类模型可以看作是一个黑箱，当给定未知记录的属性集上的值时，它自动地赋予未知样本类标号。

输入属性集(x) ⟹ 分类模型 ⟹ 输入类标号(y)

图 5-1 分类器的任务是根据输入属性集 x 确定类标号 y

例如，假设有一种叫做毒蜥的生物，其特征如表 5-2 所示。

表 5-2 毒蜥的特征数据

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
毒蜥	冷血	鳞片	否	否	否	是	是	?

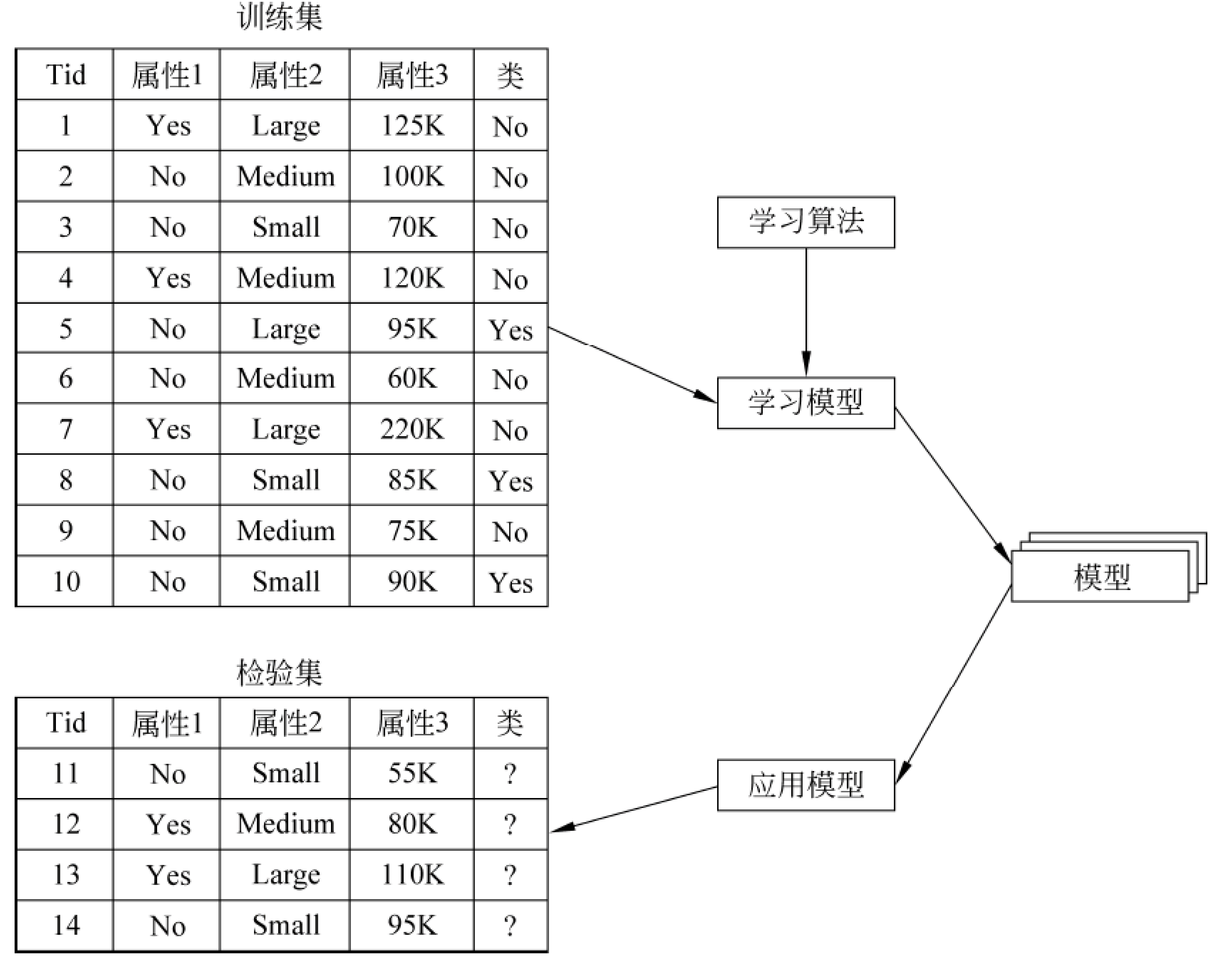
可以根据表 5-1 中的数据集建立的分类模型来确定该生物所属的类。

假设销售经理希望预测一位给定的顾客一次购物将花多少钱，这个数据分析任务就是**数值预测**(numeric prediction)的一个例子，其中所构造的模型预测一个连续值函数或有序值而不是类标号。这种模型是**预测器**(predictor)。**回归分析**(regression analysis)是数值预测最常用的统计方法，因此这两个术语常常作为同义词使用，尽管还存在其他数值预测方法。分类和数值预测是**预测问题**的两种主要类型。

5.1.2 解决分类问题的一般方法

分类技术(或分类法)是一种根据输入数据集建立分类模型的系统方法。分类法包括决策树分类法、基于规则的分类法、神经网络、支持向量机和朴素贝叶斯分类法。这些技术都使用一种**学习算法**(learning algorithm)确定分类模型,该模型能够很好地拟合输入数据中类标号和属性集之间的联系。学习算法得到的模型不仅要很好地拟合输入数据,还要能够正确地预测未知样本的类标号。因此,训练算法的主要目标就是建立具有很好的泛化能力的模型,即建立能够准确地预测未知样本类标号的模型。

图 5-2 展示了解决分类问题的一般方法。首先,需要一个**训练集**(training set),它由类标号已知的记录组成。使用训练集建立分类模型,该模型随后将运用于**检验集**(test set),检验集由类标号未知的记录组成。



训练集

Tid	属性1	属性2	属性3	类
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

检验集

Tid	属性1	属性2	属性3	类
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?

图 5-2 建立分类模型的一般方法

由于提供了每个训练元组的类标号,这一阶段也称为**监督学习**(supervised learning,即分类器的学习在被告知每个训练元组属于哪个类的“监督”下进行的)。**无监督学习**(unsupervised learning,或称聚类)则不同,每个训练元组的类标号是未知的,并且要学习的类的个数或集合也可能事先不知道。例如,如果没有用于训练集的数据,则可以使用聚类尝试确定“相似元组的组群”。

分类过程的学习模型也可以看作是学习一个映射或函数 $y=f(X)$,它可以预测给定元组 X 的类标号 y。在这种观点下,我们希望学习把数据类分开的映射或函数。在典型情况下,该映射用分类规则、决策树或数学公式的形式提供。

在应用模型阶段，使用模型进行分类，首先要评估分类器的预测准确率。如果使用训练集来衡量分类器的准确率，则评估可能是乐观的，因为分类器趋向于**过分拟合**(overfit)该数据(即在学习期间，它可能包含了训练数据中的某些特定的异常，这些异常不在一般数据集中出现)。因此，需要使用由检验元组和与它们相关联的类标号组成的**检验集**，它们独立于训练元组，即不使用它们构造分类器。

分类器在给定检验集上的**准确率**(accuracy)是分类器正确分类的检验元组所占的百分比。每个检验元组的类标号与学习模型对该元组的类预测进行比较。如果认为分类器的准确率是可以接受的，那么就可以用它对类标号未知的数据元组进行分类(这种数据在机器学习中也称为“未知的”或“先前未见到的”数据)。

5.2 决策树算法

本节介绍决策树分类法，这是一种简单但广泛使用的分类技术。在 5.2.1 节通过案例对决策树归纳过程进行介绍。决策树的建立过程在 5.2.2 节给出。5.2.3 节和 5.2.4 节分别给出了属性测试条件的方法和选择最佳划分的度量的方法。5.2.5 节给出决策树归纳算法。树剪枝的概念在 5.2.6 节介绍。5.2.7 节对决策树归纳的特点做了总结。

5.2.1 决策树归纳

决策树归纳是从有类标号的训练元组中学习决策树。**决策树**(decision tree)是一种类似于流程图的树结构，其中，每个**内部节点**(internal node，即非树叶节点)表示在一个属性上的测试，每个分枝代表该测试的一个输出，而每个**树叶节点**(leaf node)(或终端节点)存放一个类标号。树的最顶层节点是**根节点**(root node)。叶节点用矩形表示，而内部节点和根节点用椭圆表示。有些决策树算法只产生二叉树，而另一些决策树算法可能产生非二叉的树。例如，在图 5-3 中，在根节点处，使用体温这个属性把冷血脊椎动物和恒温脊椎动物区别开来。因为所有的冷血脊椎动物都是非哺乳动物，所以用一个类标号为“非哺乳动物”的叶节点作为根节点的右孩子。如果脊椎动物的体温是恒温的，则接下来用“胎生”这个属性来区分哺乳动物与其他恒温动物。

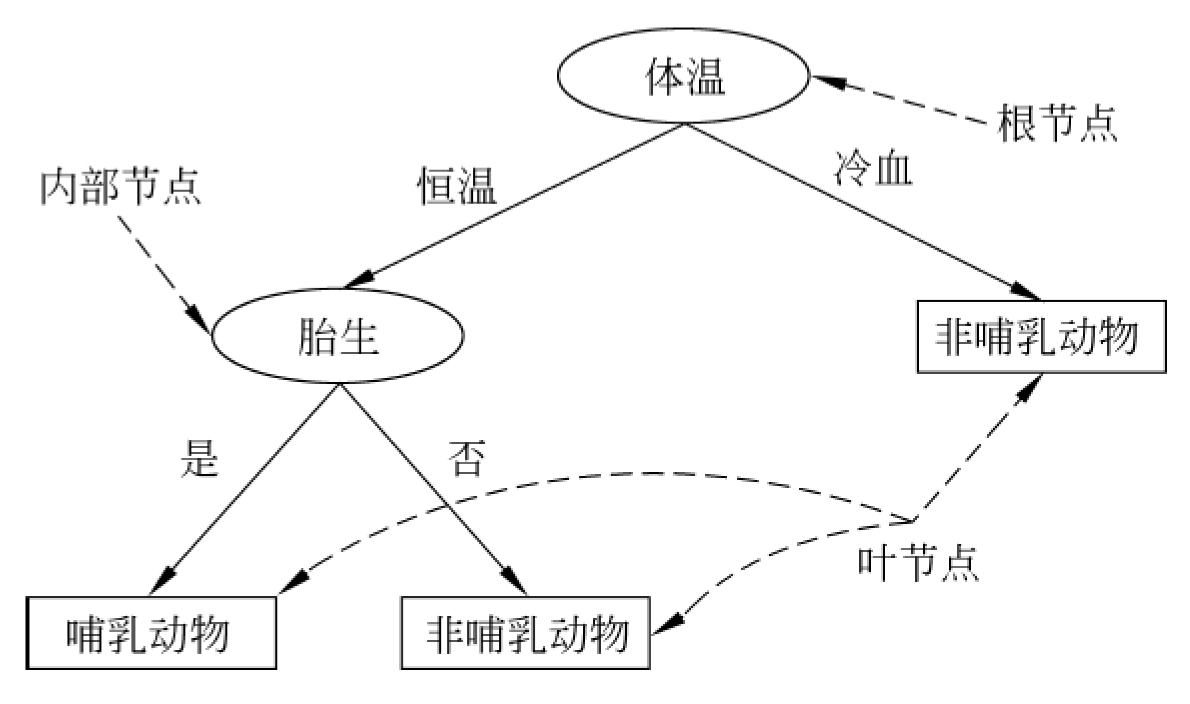
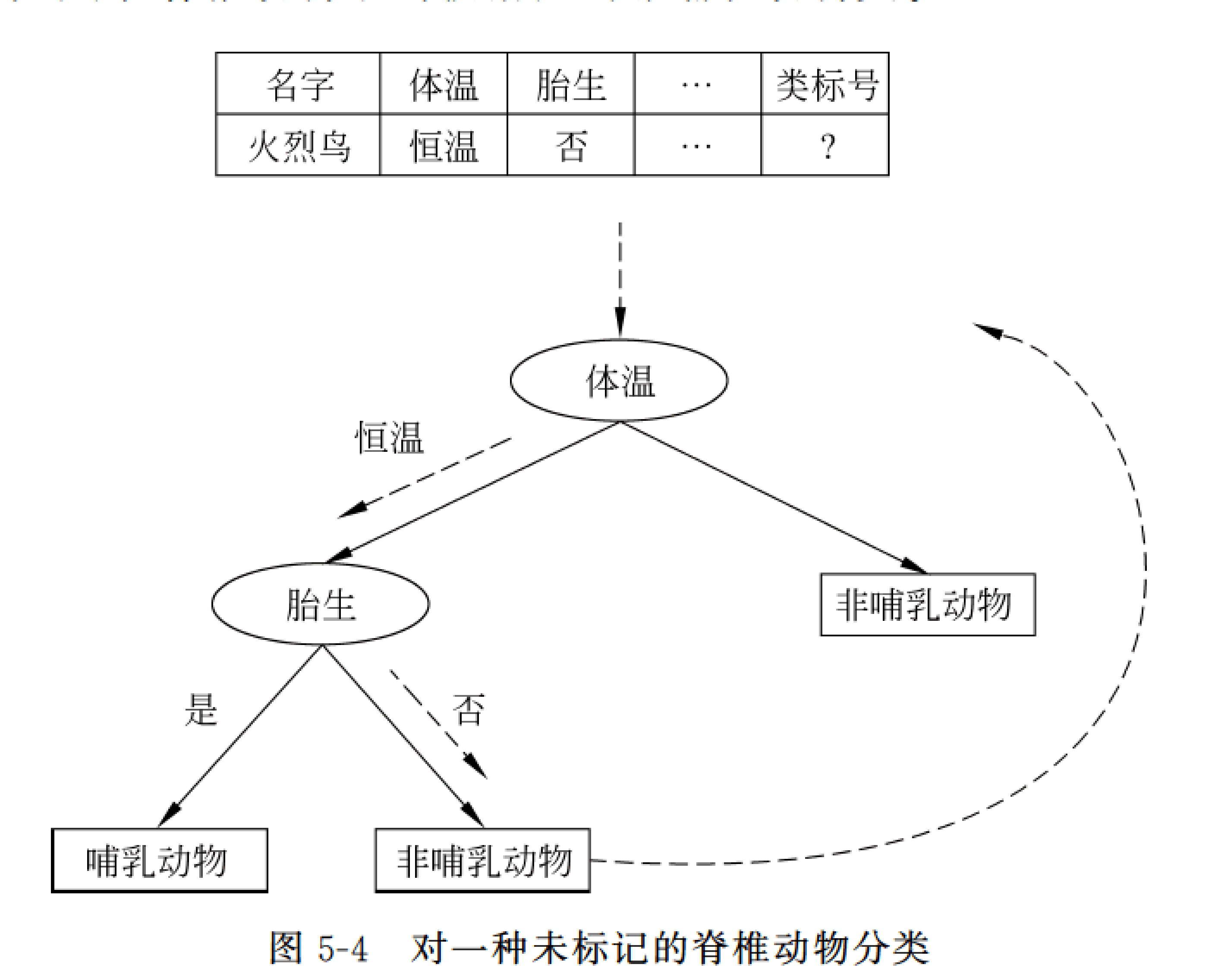


图 5-3　哺乳动物分类问题的决策树

给定一个类标号未知的元组 X，在决策树上测试该元组的属性值。跟踪一条由根到叶节点的路径，该叶节点就存放着该元组的类预测。决策树容易转换成分类规则。

决策树分类器的构造不需要任何领域知识或参数设置，因此适合于探测式知识发现。决策树可以处理高维数据。获取的知识用树的形式表示是直观的，并且容易被人理解。决策树归纳的学习和分类步骤是简单和快速的。一般而言，决策树分类器具有很高的准确率。然而，成功的使用可能依赖手头的数据。决策树归纳算法已经成功地应用于许多领域的分类，如医学、制造和生产、金融分析、天文学和分子生物学。决策树是许多商业规则归纳系统的基础。

一旦构造了决策树，对检验记录进行分类就相当容易了。从树的根节点开始，将测试条件用于检验记录，根据测试结果选择适当的分枝。沿着该分枝或者到达另一个内部节点，使用新的测试条件，或者到达一个叶节点。到达叶节点之后，叶节点的类标号就被赋值给该检验记录。例如，图 5-4 显示了应用决策树预测火烈鸟的类标号所经过的路径，路径终止于类标号为“非哺乳动物”的叶节点。虚线表示在未标记的脊椎动物上使用各种属性测试条件的结果，该脊椎动物最终被指派到非哺乳动物类。



名字	体温	胎生	…	类标号
火烈鸟	恒温	否	…	?

图 5-4　对一种未标记的脊椎动物分类

5.2.2　决策树构建

原则上讲，对于给定的属性集，可以构造的决策树的数目达指数级。尽管某些决策树比其他决策树更准确，但是由于搜索空间是指数规模的，找出最佳决策树在计算上是不可行的。尽管如此，人们还是开发了一些有效的算法，能够在合理的时间内构造出具有一定准确率的次最优决策树。这些算法通常都采用贪心策略（即非回溯的），在选择划分数据的属性时，采取一系列局部最优决策来构造决策树，Hunt 算法就是一种这样的算法。Hunt 算法是许多决策树算法的基础，包括 ID3、C4.5 和 CART。

在 Hunt 算法中，通常将训练记录相继划分成较纯的子集，以递归方式建立决策树。

设 D_t 是与节点 t 相关联的训练记录集，而 $y=\{y_1, y_2, \cdots, y_c\}$ 是类标号，Hunt 算法的递归定义如下：

(1) 如果 D_t 中所有记录都属于同一个类 y_t，则 t 是叶节点，用 y_t 标记。

(2) 如果 D_t 中包含属于多个类的记录，则选择一个**属性测试条件**(attribute test condition)，将记录划分成较小的子集。对于测试条件的每个输出，创建一个子女节点，并根据测试结果将 D_t 中的记录发布到子女节点中。然后，对于每个子女节点，递归地调用该算法。

为了解释该算法如何执行，考虑如下问题：预测贷款申请者是会按时归还贷款还是会拖欠贷款。对于这个问题，训练数据集可以通过考察以前贷款者的贷款记录来构造。在表 5-3 所示的例子中，每条记录都包含贷款者的个人信息以及贷款者是否拖欠贷款的类标号。

表 5-3　训练数据集：预测拖欠银行贷款的贷款者

Tid	有房者	婚姻状况	年收入	拖欠贷款者
1	是	单身	125k	否
2	否	已婚	100k	否
3	否	单身	70k	否
4	是	已婚	120k	否
5	否	离异	95k	是
6	否	已婚	60k	否
7	是	离异	220k	否
8	否	单身	85k	是
9	否	已婚	75k	否
10	否	单身	90k	是

该分类问题的初始决策树只有一个节点，类标号为“拖欠贷款者＝否”(见图 5-5(a))，意味着大多数贷款者都按时归还贷款。然而，该树需要进一步的细化，因为根节点包含两个类的记录。根据“有房者”测试条件，这些记录被划分为较小的子集，如图 5-5(b)所示。选取属性测试条件的理由稍后讨论，目前，假定此处这样选是划分数据的最优选择。接下来，对根节点的每个子女递归地调用 Hunt 算法。从表 5-3 给出的训练数据集可以看出，有房的贷款者都按时偿还了贷款，因此，根节点的左子女为叶节点，标记为“拖欠贷款者＝否”(见图 5-5(b))。对于右子女，需要继续递归调用 Hunt 算法，直到所有的记录都属于同一个类为止。每次递归调用所形成的决策树显示在图 5-5(c)和图 5-5(d)中。

如果属性值的每种组合都在训练数据中出现，并且每种组合都具有唯一的类标号，则 Hunt 算法是有效的，但是对于大多数实际情况，这些假设太苛刻了，因此，需要附加的条件来处理以下的情况。

(1) 算法的第二步(图 5-5(b))所创建的子女节点可能为空，即不存在与这些节点相

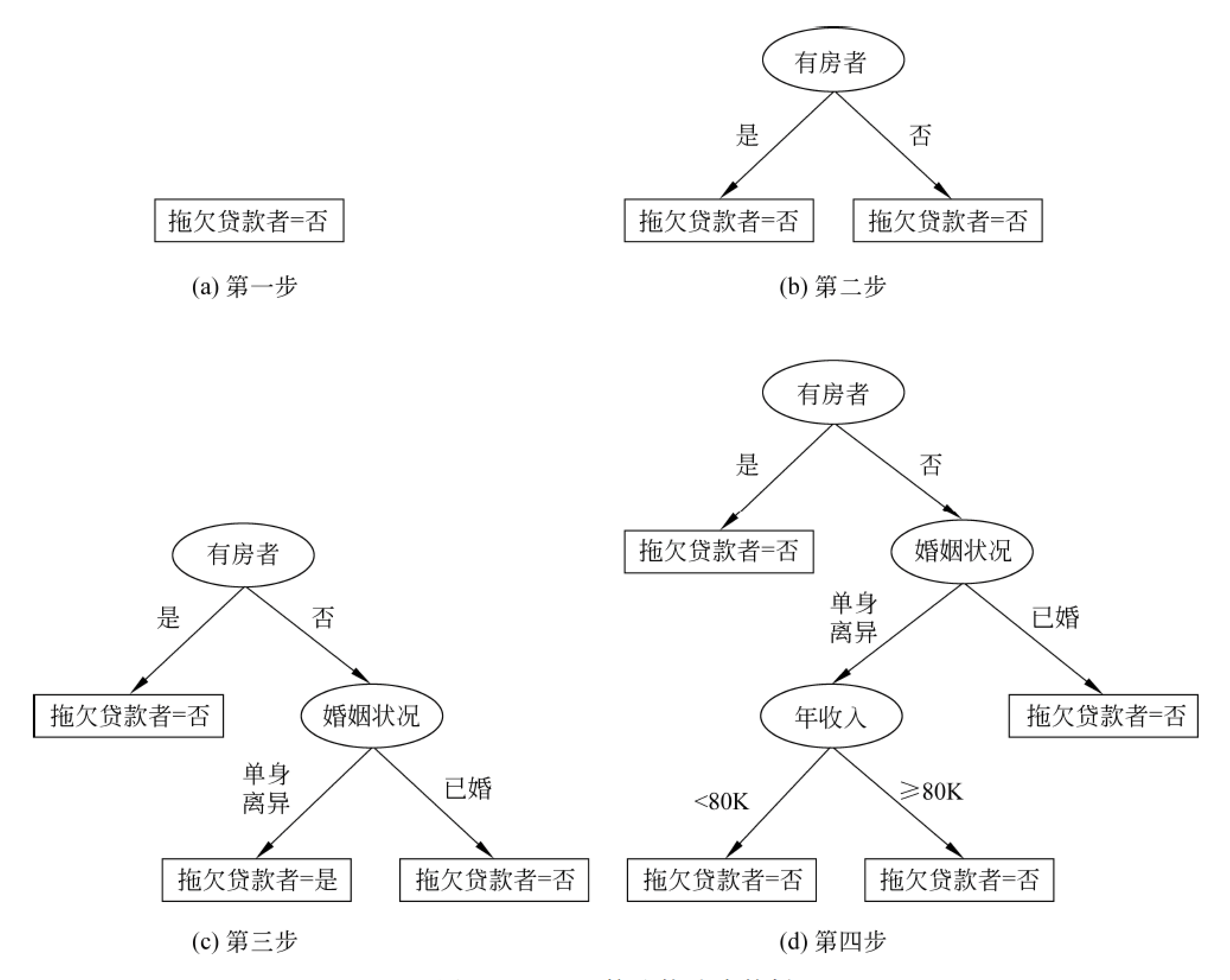


图 5-5　Hunt 算法构造决策树

关联的记录。如果没有一个训练记录包含与这样的节点相关联的属性值组合，这种情形就可能发生。这时，该节点成为叶节点，类标号为其父节点上训练记录中的多数类。

（2）在第二步，如果与相关联的所有记录都具有相同的属性值（目标属性除外），则不可能进一步划分这些记录。在这种情况下，该节点为叶节点，其标号为与该节点相关联的训练记录中的多数类。

决策树归纳的学习算法必须解决下面两个问题。

（1）**如何分裂训练记录？** 树增长过程的每个递归步都必须选择一个属性测试条件，将记录划分成较小的子集。为了实现这个步骤，算法必须提供为不同类型的属性指定测试条件的方法，并且提供每个测试条件的客观度量。

（2）**如何停止分裂过程？** 需要有结束条件，以终止决策树的生长过程。一个可能的策略是分裂节点，直到所有的记录都属于同一个类，或者所有的记录都具有相同的属性值。尽管每个结束条件对于结束决策树归纳算法都是充分的，但是还可以使用其他的标准提前终止树的生长过程。

5.2.3　属性测试条件的表示方法

决策树归纳算法必须为不同类型的属性提供表示属性测试条件和其对应输出的

方法。

（1）**二元属性**。二元属性的测试条件产生两个可能的输出，如图 5-6 所示。

（2）**标称属性**。由于标称属性有多个属性值，它的测试条件可以用两种方法表示，如图 5-7 所示。对于多路划分（图 5-7(a)），其输出数取决于该属性不同属性值的个数。例如，如果属性婚姻状况有 3 个不同的属性值——单身、已婚、离异，则它的测试条件就会产生一个三路划分。另一方面，某些决策树算法（如 CART）只产生二元划分，它们考虑创建 k 个属性值的二元划分的所有 $2^{k-1}-1$ 种方法。图 5-7(b)显示了把婚姻状况的属性值划分为两个子集的 3 种不同的分组方法。

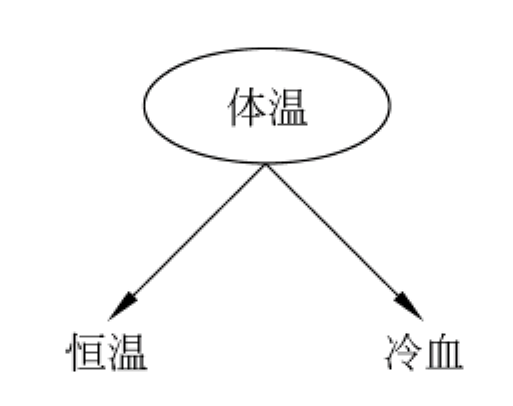


图 5-6　二元属性的测试条件

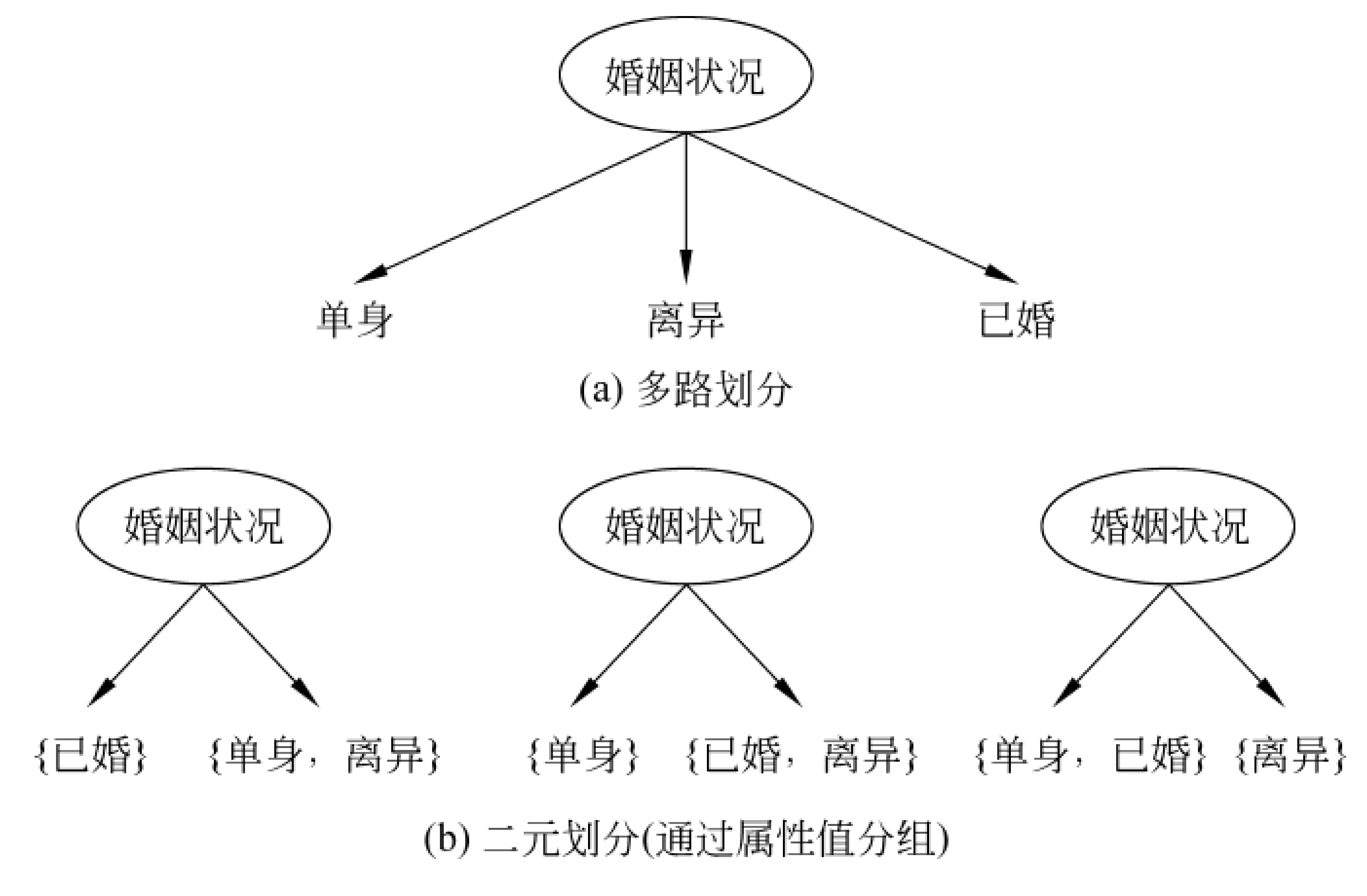


图 5-7　标称属性的测试条件

（3）**序值属性**。序值属性也可以产生二元或多路划分，只要不违背序值属性值的有序性，就可以对属性值进行分组。图 5-8 显示了按照属性衬衣尺码划分训练记录的不同的方法。图 5-8(a)和图 5-8(b)中的分组保持了属性值间的序关系，而图 5-8(c)所示的分组则违反了这一性质，因为它把小号和大号分为一组，把中号和加大号放在另一组。

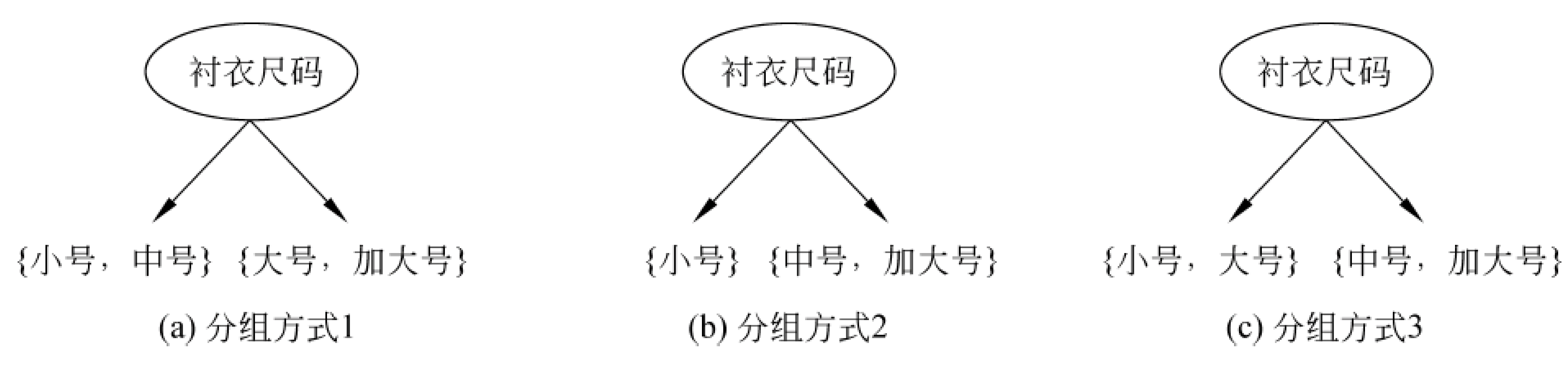


图 5-8　序值属性分组的不同方式

（4）**连续属性**。对于连续属性来说，测试条件可以是具有二元输出的比较测试（$A<v$）或（$A\geqslant v$），也可以是具有形如 $v_i\leqslant A<v_{i+1}$（$i=1,2,\cdots,n$）输出的范围查询。图 5-9 显示了这些方法的差别。对于二元划分，决策树算法必须考虑所有可能的划分点 v，并从中选择产生最佳划分的点。

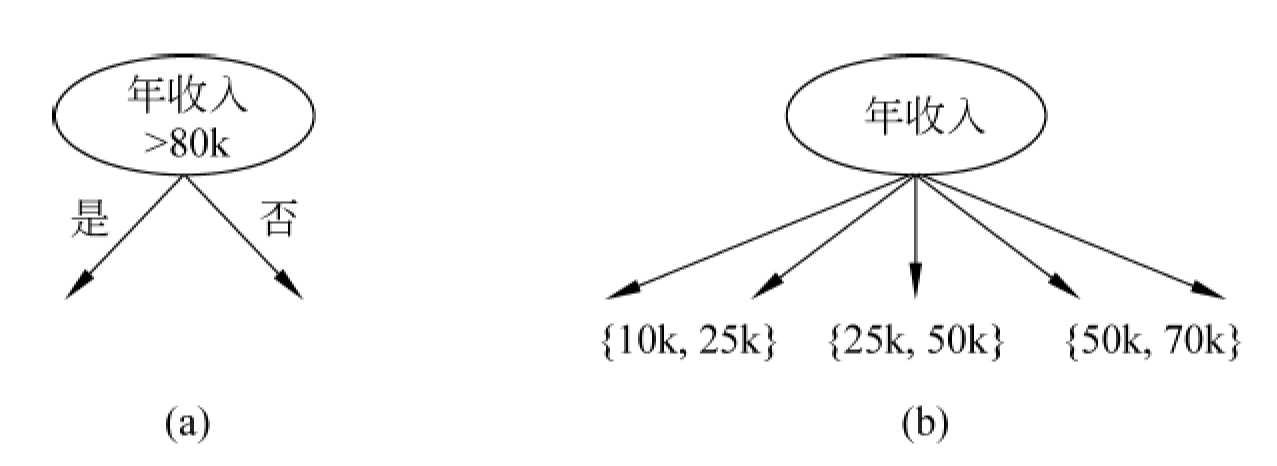


图 5-9 连续属性的测试条件

对于多路划分，算法必须考虑所有可能的连续值区间。可以采用离散化的策略，离散化之后，每个离散化区间赋予一个新的序数值，只要保持有序性，相邻的值还可以聚集成较宽的区间。

5.2.4 选择最佳划分的度量

有很多度量可以用来确定划分记录的最佳方法，这些度量用划分前和划分后记录的类分布定义。

设 $p(i|t)$ 表示给定节点 t 中属于类 i 的记录所占的比例，有时，省略节点 t，直接用 p_i 表示该比例。在二元分类问题中，任意节点的类分布都可以记作 (p_0, p_1)，其中 $p_1 = 1 - p_0$。例如，考虑图 5-10 中的测试条件，划分前的类分布是(0.5,0.5)，因为来自每个类的记录数相等。如果使用性别属性来划分数据，则子女节点的类分布分别为(0.6,0.4)和(0.4,0.6)，虽然划分后两个类的分布不再平衡，但是子女节点仍然包含两个类的记录；按照第二个属性车型进行划分，将得到纯度更高的划分。

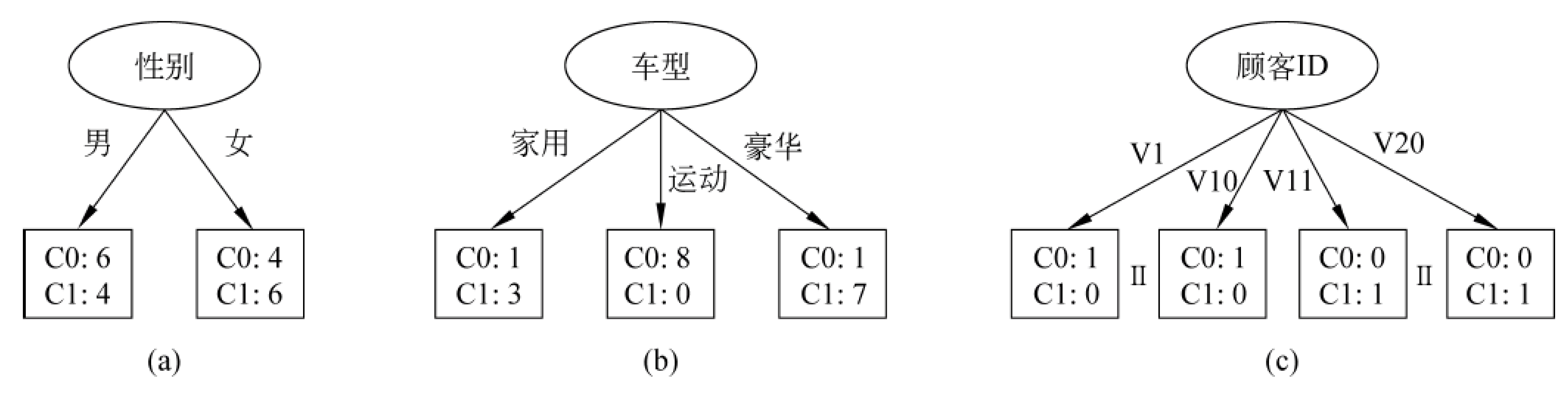


图 5-10 多路划分与二元划分

选择最佳划分的度量通常是根据划分后子女节点不纯的程度。不纯的程度越低，类分布就越倾斜。例如，类分布为(0,1)的节点具有零不纯性，而均衡分布(0.5,0.5)的节点具有最高的不纯性。不纯性度量的例子如下：

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i \mid t)\log_2 p(i \mid t) \tag{5-1}$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i \mid t)]^2 \tag{5-2}$$

$$\text{Error}(t) = 1 - \max_i [p(i \mid t)] \tag{5-3}$$

其中 c 是类的个数，并且在计算熵时，$O(\log_2 0) = 0$。

图 5-11 显示了二元分类问题不纯性度量值的比较，p 表示属于其中一个类的记录所占的比例。从图中可以看出，3 种方法都在类分布均衡时(即当 $p=0.5$ 时)达到最大值，而当所有记录都属于同一个类时(p 等于 1 或 0)达到最小值。下面给出 3 种不纯性度量方法的计算实例。

节点 N_1	计数
类=0	0
类=1	6

$\text{Gini}=1-(0/6)^2-(6/6)^2=0$

$\text{Entropy}=-(0/6)\log_2(0/6)-(6/6)\log_2(6/6)=0$

$\text{Error}=1-\max[0/6,6/6]=0$

节点 N_2	计数
类=0	1
类=1	5

$\text{Gini}=1-(1/6)^2-(5/6)^2=0.278$

$\text{Entropy}=-(1/6)\log_2(1/6)-(5/6)\log_2(5/6)=0.650$

$\text{Error}=1-\max[1/6,5/6]=0.167$

节点 N_3	计数
类=0	3
类=1	3

$\text{Gini}=1-(3/6)^2-(3/6)^2=0.5$

$\text{Entropy}=-(3/6)\log_2(3/6)-(3/6)\log_2(3/6)=1$

$\text{Error}=1-\max[3/6,3/6]=0.5$

从上面的例子及图 5-11 可以看出，不同的不纯性度量是一致的。根据计算，节点 N_1 具有最低的不纯性度量值，接下来依次是 N_2、N_3。虽然结果是一致的，但是作为测试条件的属性选择仍然因不纯性度量的选择而异。

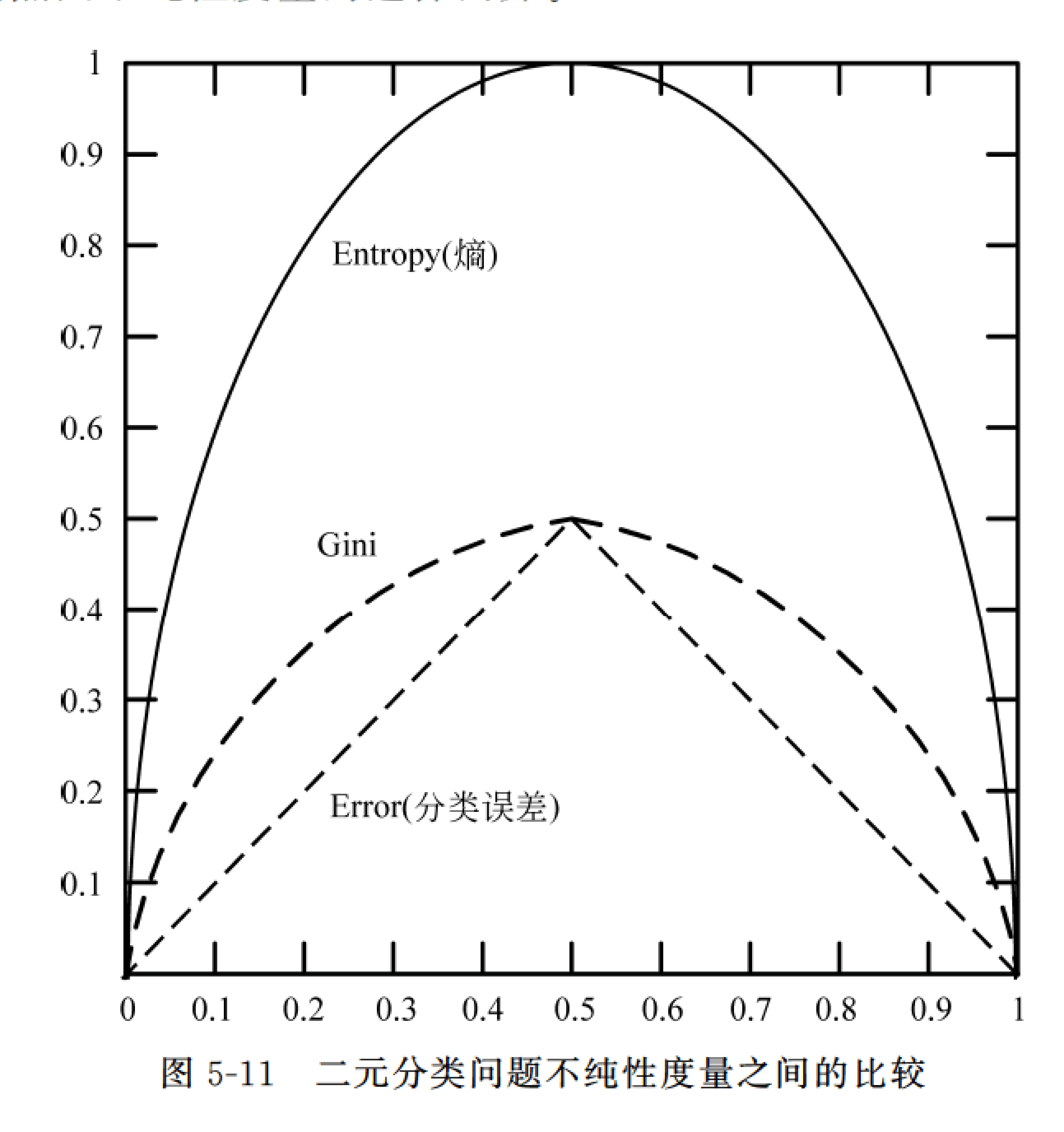


图 5-11　二元分类问题不纯性度量之间的比较

为了确定测试条件的效果，需要比较父节点（划分前）的不纯程度和子女节点（划分后）的不纯程度，它们的差越大，测试条件的效果就越好。增益 Δ 是一种可以用来确定划分效果的标准：

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) \tag{5-4}$$

其中，$I(\text{parent})$是给定节点的不纯性度量，N 是父节点上的记录总数，k 是属性值的个数，$N(v_j)$是与子女节点 v_j 相关联的记录个数。决策树归纳算法通常选择最大化增益 Δ 的测试条件，因为对所有的测试条件来说，$I(\text{parent})$是一个不变的值，所以最大化增益等价于最小化子女节点的不纯性度量的加权平均值。最后，当选择熵（Entropy）作为式（5-4）的不纯性度量时，熵的差就是所谓**信息增益**（information gain），用 Δ_{info} 表示。

1. 二元属性的划分

考虑图 5-12 中的图表，假设有两种方法将数据划分成较小的子集。划分前，Gini 指标等于 0.5，因为属于两个类的记录个数相等。如果选择属性 A 划分数据，节点 N_1 的 Gini 指标等于 0.4898，而 N_2 的 Gini 指标等于 0.480，派生节点的 Gini 指标的加权平均为(7/12)×0.4898+(5/12)×0.480=0.486。类似地，可以计算出属性 B 的 Gini 指标加权平均是 0.371。因为属性 B 具有更小的 Gini 指标，它比属性 A 更可取。

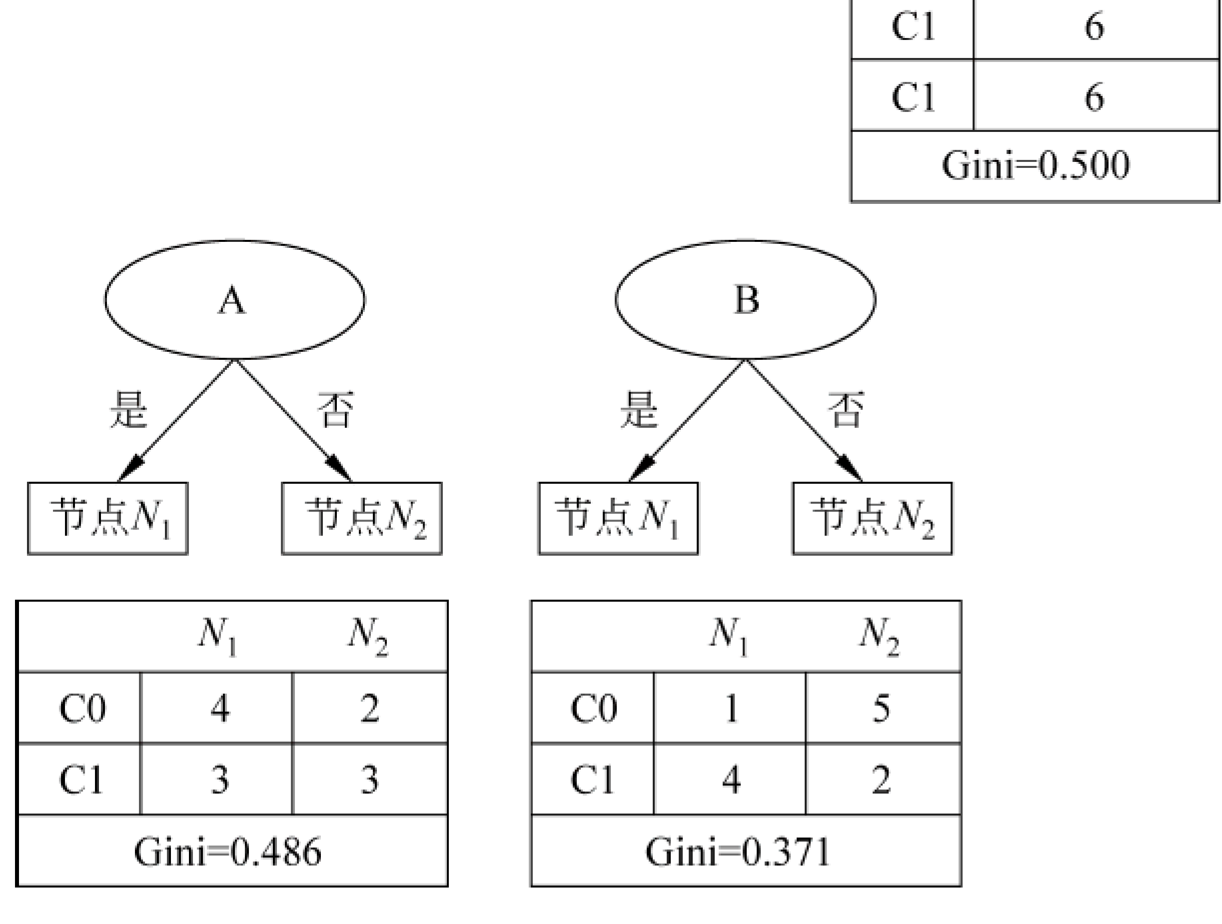


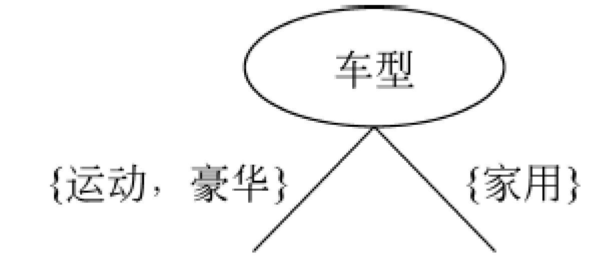
图 5-12 划分二元属性

2. 标称属性的划分

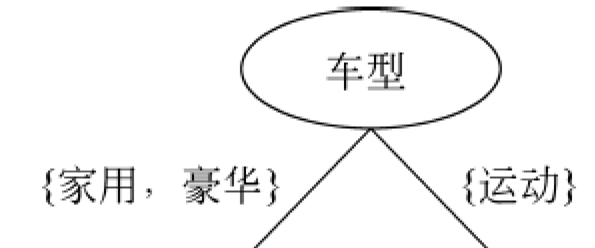
正如前面强调的，标称属性可以产生二元划分或多路划分，如图 5-13 所示。二元划分的 Gini 指标的计算与二元属性类似。对于车型属性第一种二元分组，{运动，豪华}的 Gini 指标是 0.4922，而{家用}的 Gini 指标是 0.3750。该分组的 Gini 指标加权平均是

16/20×0.4922+4/20×0.3750=0.468

类似地，对第二种二元分组{运动}和{家用，豪华}，Gini 指标加权平均是 0.167。第



	车型	
	{运动，豪华}	{家用}
C0	9	1
C1	7	3
Gini	0.468	



	车型	
	{运动}	{家用，豪华}
C0	8	2
C1	0	10
Gini	0.167	



	车型		
	{家用}	{运动}	{豪华}
C0	1	8	1
C1	3	0	7
Gini	0.163		

图 5-13　划分标称属性

二种分组的 Gini 指标值相对较低，因为其对应的子集的纯度高得多。

对于多路划分，需要计算每个属性值的 Gini 指标。Gini({家用})＝0.375，Gini({运动})＝0，Gini({豪华})＝0.219，多路划分的总 Gini 指标为

$$4/20\times0.375+8/20\times0+8/20\times0.219=0.163$$

多路划分的 Gini 指标比两个二元划分都小。这一结果并不奇怪，因为二元划分实际上合并了多路划分的某些输出，自然降低了子集的纯度。

3. 连续属性的划分

考虑图 5-14 的例子，其中测试条件“年收入≤y”用来划分拖欠贷款分类问题的训练记录。用穷举方法确定 v 的值，将 N 个记录中所有的属性值都作为候选划分点。对每个候选 v 都要扫描一次数据集，统计年收入大于 v 和小于等于 v 的记录数，然后计算每个候选的 Gini 指标，并从中选择具有最小值的候选划分点。这种方法的计算代价是昂贵的，因为对每个候选划分点计算 Gini 指标需要 $O(N)$ 次操作，由于有 N 个候选，总的计算复杂度为 $O(N^2)$。为了降低计算复杂度，按照年收入将训练记录排序，所需要的时间为 $O(N\log N)$，从两个相邻的排过序的属性值中选择中间值作为候选划分点，得到候选划分点 55、65、72 等。无论如何，与穷举方法不同，在计算候选划分点的 Gini 指标时，不需考察所有 N 个记录。

类		No		No		No		Yes		Yes		Yes		No		No		No		No		
	年收入																					
		60		70		75		85		90		95		100		120		125		220		
	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.400		0.420	

图 5-14　连续属性的划分

对第一个候选 $v=55$，没有年收入小于或等于 55k 的记录，所以年收入小于 55k 的派生节点的 Gini 指标是 0。另一方面，年收入大于 55k 的样本记录数为 3(类 Yes)和 7(类 No)，这样，该节点的 Gini 指标是 0.420。该候选划分的总 Gini 指标为 $0\times0+1\times0.420=0.420$。

对第二个候选 $v=65$，通过更新上一个候选的类分布，就可以得到该候选的类分布。更具体地说，新的分布通过考察具有最低年收入(即 60k)的记录的类标号得到。因为该记录的类标号是 No，所以类 No 的数量从 0 增加到 1(对于年收入≤65k)，和从 7 降到 6(对于年收入>65k)，类 Yes 的分布保持不变。新的候选划分点的加权平均 Gini 指标为 0.400。

重复这样的计算，直到算出所有候选的 Gini 指标值，如图 5-14 所示。最佳的划分点对应于产生最小 Gini 指标值的点，即 $v=97$。该过程代价相对较低，因为更新每个候选划分点的类分布所需的时间是一个常数。该过程还可以进一步优化：仅考虑位于具有不同类标号的两个相邻记录之间的候选划分点。例如，因为前 3 个排序后的记录(年收入分别为 60k、70k 和 75k)具有相同的类标号，所以最佳划分点肯定不会在 60k 和 75k 之间，因此，候选划分点 $v=$55k，65k，72k，87k，92k，110k，122k，172k，230k 都将被忽略，因为它们都位于具有相同类标号的相邻记录之间。该方法使得候选划分点的个数从 11 个降到 2 个。

4. 增益率

熵和 Gini 指标等不纯性度量趋向有利于具有大量不同值的属性。图 5-10 显示了 3 种可供选择的测试条件，划分本章习题第 7 题中的数据集。第一个测试条件“性别”与第二个测试条件“车型”相比，容易看出“车型”似乎提供了更好的划分数据的方法，因为它产生更纯的划分，但“顾客 ID”不是一个有预测性的属性，因为每个样本在该属性上的值都是唯一的。即使在不太极端的情形下，也不会希望产生大量输出的测试条件，因为与每个划分相关联的记录太少，以致不能做出可靠的预测。

解决该问题的策略有两种。一种策略是限制测试条件只能是二元划分，CART 这样的决策树算法采用的就是这种策略；另一种策略是修改评估划分的标准，把属性测试条件产生的输出数也考虑进去，例如，决策树算法 C4.5 采用称作增益率(gain ratio)的划分标准来评估划分。增益率定义如下：

$$\text{Gain_ratio}=\frac{\Delta_{\text{info}}}{\text{Split_Info}} \tag{5-5}$$

其中，划分信息 $\text{Split_Info}=-\sum_{i=1}^{k}P(v_i)\log_2P(v_i)$，而 k 是划分的总数。例如，如果每个属性值具有相同的记录数，则 $\forall i: P(v_i)=1/k$，而划分信息等于 $\log_2k$。这说明如果某个属性产生了大量的划分，它的划分信息将会很大，从而降低了增益率。

5.2.5 决策树归纳算法

算法 5.1 给出了称作 TreeGrowth 的决策树归纳算法的框架。该算法的输入是训练记录集 E 和属性集 F。算法递归地选择最优的属性来划分数据(步骤 7)，并扩展树的叶

节点(步骤 11 和步骤 12),直到满足结束条件(步骤 1)。算法的细节如下:

(1) 函数 createdNode()为决策树建立新的节点。决策树的节点或者是一个测试条件,记作 node. test_cond,或者是一个类标号,记作 node. label。

(2) 函数 find_best_split()确定应当选择哪个属性作为划分训练记录的测试条件。如前所述,测试条件的选择取决于使用哪种不纯性度量来评估划分,一些广泛使用的度量包括熵、Gini 指标和 χ^2 统计量。

(3) 函数 classify()为叶节点确定类标号。对于每个叶节点 t,$p(i|t)$表示该节点上属于类 i 的训练记录所占的比例,在大多数情况下,都将叶节点指派到具有多数记录的类:

$$\text{leaf. label} = \arg\max_i p(i \mid t) \tag{5-6}$$

其中,操作 arg max 返回最大值 $p(i|t)$的参数值 i。$p(i|t)$除了确定叶节点类标号所需要的信息之外,还可以用来估计分配到叶节点 t 的记录属于类 i 的概率。

(4) 函数 stopping_cond()检查是否所有的记录都属于同一个类,或者都具有相同的属性值,决定是否终止决策树的增长。终止递归函数的另一种方法是检查记录数是否小于某个最小阈值。

算法 5-1 决策树归纳算法的框架

```
TreeGrowth(E,F)
1: if stopping_cond(E,F) = true then
2:   leaf = createNode()
3:   leaf.label = Classify(E)
4:   return leaf
5: else
6:   root = createNode()
7:   root.test_cond = find_best_split(E,F)
8:   令 V={v|v 是 root.test_cond 的一个可能的输出}
9:   for 每个 v∈V do
10: E_v ={e/root.test_cond(e) =v 并且 e∈E}
11:   child = TreeGrowth(E_v,F)
12:   将 child 作为 root 的派生节点添加到树中,并将边(root→child)标记为 v
13: end for
14: end if
15: return root
```

建立决策树之后,可以进行**树剪枝**(tree-pruning),以减小决策树的规模。决策树过大容易受所谓**过分拟合**(overfitting)现象的影响。通过修剪初始决策树的分枝,剪枝有助于提高决策树的泛化能力。

5.2.6 树剪枝

在决策树创建时,由于数据中的噪声和离群点,许多分枝反映的是训练数据中的异常。剪枝方法用来处理这种过分拟合数据问题。通常,这种方法使用统计度量剪掉最不可

靠的分枝。一棵未剪枝的树和它剪枝后的版本显示在图 5-15 中。剪枝后的树更小，更简单，因此更容易理解。通常，它们在正确地对独立的检验集分类时比未剪枝的树更快更好。

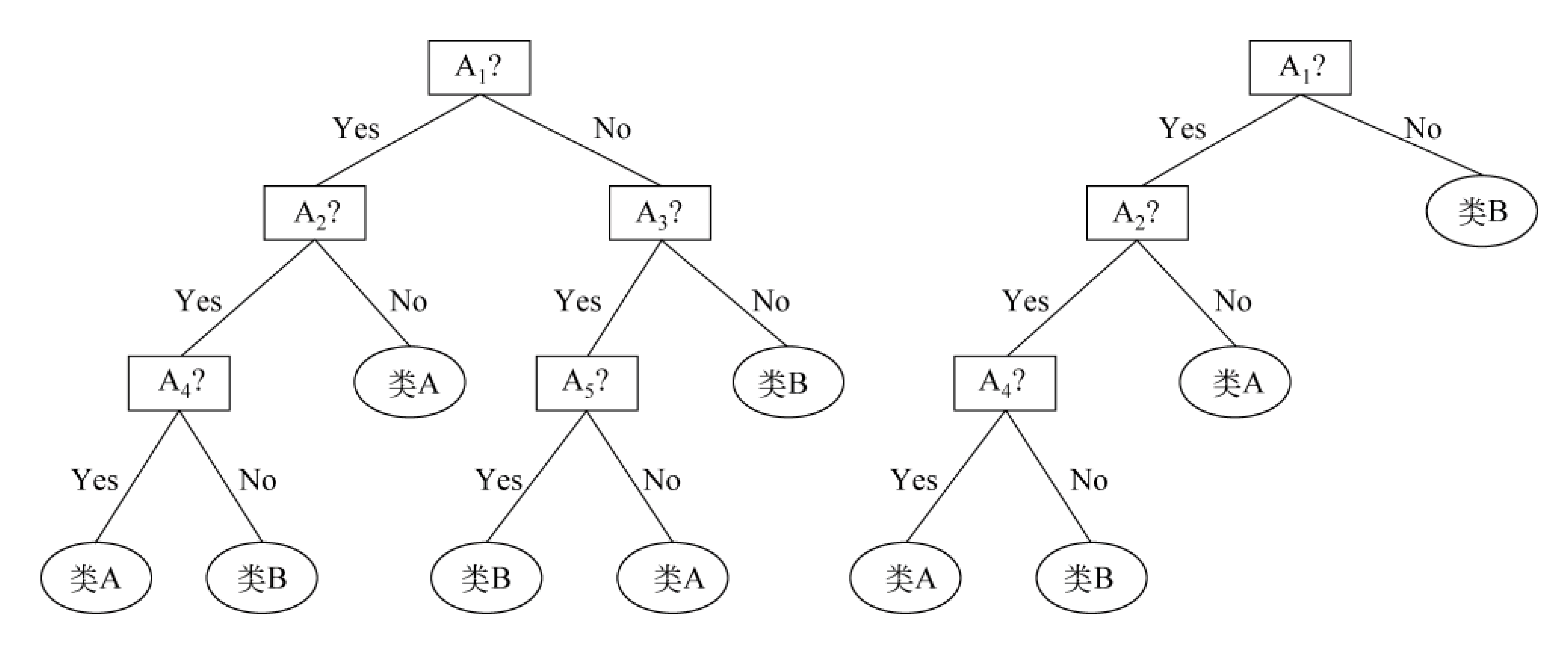


图 5-15 一棵未剪枝的决策树和它剪枝后的版本

有两种常用的剪枝方法：先剪枝和后剪枝。

在**先剪枝**(prepruning)方法中，通过提前停止树的构建(例如，通过决定在给定的节点不再分裂或划分训练元组的子集)而对树剪枝。一旦停止，节点就称为树叶。该树叶可以持有子集元组中最频繁的类或这些元组的概率分布。

在构造树时，可以使用诸如统计显著性、信息增益、Gini 指数等度量来评估划分的优劣。如果划分一个节点的元组导致低于预定义阈值的划分，则给定子集的进一步划分将停止。然而，选取一个适当的阈值是困难的。高阈值可能导致过分简化的树，而低阈值可能使得树的简化太少。

后剪枝(postpruning)是更常用的方法。它由“完全生长”的树剪去子树，通过删除节点的分枝，并用树叶替换它而剪掉给定节点的子树，树叶用被替换的子树中最频繁的类标记。例如，注意图 5-15 中未剪枝树的节点“A_3?”的子树。假设该子树中最频繁的类是“类B”，在剪枝的版本中，该子树被剪枝，用树叶“类 B”替换。

CART 使用的**代价复杂度**剪枝算法是后剪枝方法的一个实例。该方法把树的复杂度看做树中树叶节点的个数和树的错误率的函数(其中，**错误率**是树误分类的元组所占的百分比)。它从树的底部开始。对于每个内部节点 N，计算 N 的子树的代价复杂度和该子树剪枝后 N 的子树(即用一个树叶节点替换)的代价复杂度，比较这两个值。如果剪去节点 N 的子树导致较小的代价复杂度，则剪掉该子树；否则，保留该子树。

使用一个标记类元组的**剪枝集**来评估代价复杂度。该集合独立于用于建立未剪枝树的训练集和用于准确率评估的检验集。算法产生一个渐进的剪枝树的集合。一般而言，最小化代价复杂度的最小决策树是首选。

C4.5 算法使用一种称为**悲观剪枝**的方法，它类似于代价复杂度方法，因为它也是用错误率评估对子树剪枝做出决定。然而，悲观剪枝不需要使用剪枝集，而是使用训练集估计错误率。注意，基于训练集评估准确率或错误率过于乐观，因此具有较大的偏倚。因此，悲观

剪枝方法通过加上一个惩罚来调节从训练集得到的错误率，以抵消所出现的偏倚。

可以根据对树编码所需要的二进位位数，而不是根据估计的错误率，对树进行剪枝。“最佳”剪枝树是最小化编码二进位位数的树。这种方法采用 MDL 原则。其基本思想是：最简单的解是首选的解。与代价复杂性剪枝不同，它不需要独立的元组集。

另外，对于组合方法，先剪枝和后剪枝可以交叉使用。后剪枝所需要的计算比先剪枝多，但是通常产生更可靠的树。并未发现一种剪枝方法优于所有其他方法。尽管某些剪枝方法需要额外的数据支持，但是在处理大型数据库时，这并不是问题。

尽管剪枝后的树一般比未剪枝的树更紧凑，但是它们仍然可能很大，很复杂。决策树可能受到重复和复制的困扰，使得它们很难解释。沿着一条给定的分枝反复测试一个属性（如“age＜60?”，后面跟着“age＜45?”等）时就会出现**重复**（repetition）。**复制**（replication）是树中存在重复的子树。这些情况影响了决策树的准确率和可解释性。使用多元划分（基于组合属性的划分）可以防止该问题的出现。另一种方法是使用不同形式的知识表示（如规则）而不是决策树。

5.2.7 决策树归纳的特点

下面是对决策树归纳算法重要特点的总结。

(1) 决策树归纳是一种构建分类模型的非参数方法。换句话说，它不要求任何先验假设，不假定类和其他属性服从一定的概率分布。

(2) 找到最佳的决策树是 NP 完全问题。许多决策树算法都采取启发式的方法指导对假设空间的搜索。

(3) 已开发的构建决策树技术不需要昂贵的计算代价，即使训练集非常大，也可以快速建立模型。此外，决策树一旦建立，未知样本分类非常快，最坏情况下的时间复杂度是 $O(w)$，其中 w 是树的最大深度。

(4) 决策树相对容易解释，特别是小型的决策树。在很多简单的数据集上，决策树的准确率也可以与其他分类算法相媲美。

(5) 决策树是学习离散值函数的典型代表。然而，它不能很好地推广到某些特定的布尔问题。一个著名的例子是奇偶函数，当奇数（偶数）个布尔属性为真时其值为 0(1)。对这样的函数准确建模需要一棵具有 2^d 个节点的满决策树，其中 d 是布尔属性的个数。

(6) 决策树算法对于噪声的干扰具有相当好的鲁棒性，采用避免过分拟合的方法之后尤其如此。

(7) 冗余属性不会对决策树的准确率造成不利的影响。一个属性如果在数据中与另一个属性是强相关的，那么它是冗余的。在两个冗余属性中，如果已经选择其中一个作为用于划分的属性，则另一个将被忽略。然而，如果数据集中含有很多不相关的属性（即对分类任务没有用的属性），则某些不相关的属性可能在树的构造过程中偶然被选中，导致决策树过于庞大。通过在预处理阶段删除不相关的属性，特征选择技术能够帮助提高决策树的准确率。

(8) 由于大多数的决策树算法都采用自顶向下的递归划分方法，因此沿着树向下，记录会越来越少。在叶节点，记录可能太少，对于叶节点代表的类不能做出具有统计意义的

判决，这就是所谓的**数据碎片**(data fragmentation)问题。解决该问题的一种可行的方法是，当样本数小于某个特定阈值时停止分裂。

(9) 子树可能在决策树中重复多次，如图5-16所示，这使得决策树过于复杂，并且可能更难解释。当决策树的每个内部节点都依赖单个属性测试条件时，就会出现这种情形。由于大多数的决策树算法都采用分治划分策略，因此在属性空间的不同部分可以使用相同的测试条件，从而导致子树重复问题。

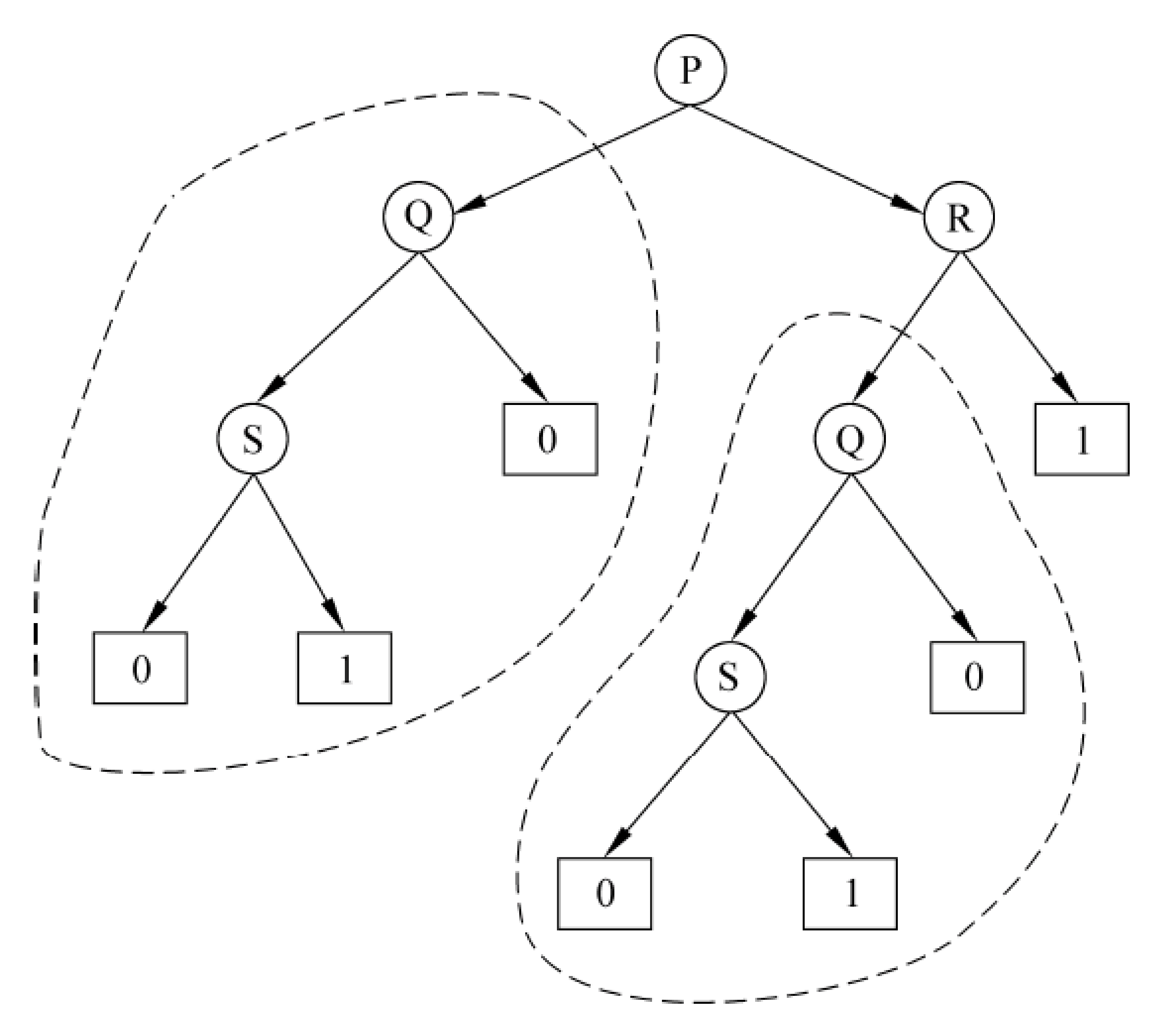


图5-16　子树重复问题

(10) 迄今为止，本章介绍的测试条件都只涉及一个属性。这样，可以将决策树的生长过程看成划分属性空间不相交的区域的过程，直到每个区域都只包含同一类的记录(见图5-17)。两个不同类的相邻区域之间的边界称作**决策边界**(decision boundary)。由于测试条件只涉及单个属性，因此决策边界是直线，即平行于"坐标轴"，这就限制了决策树

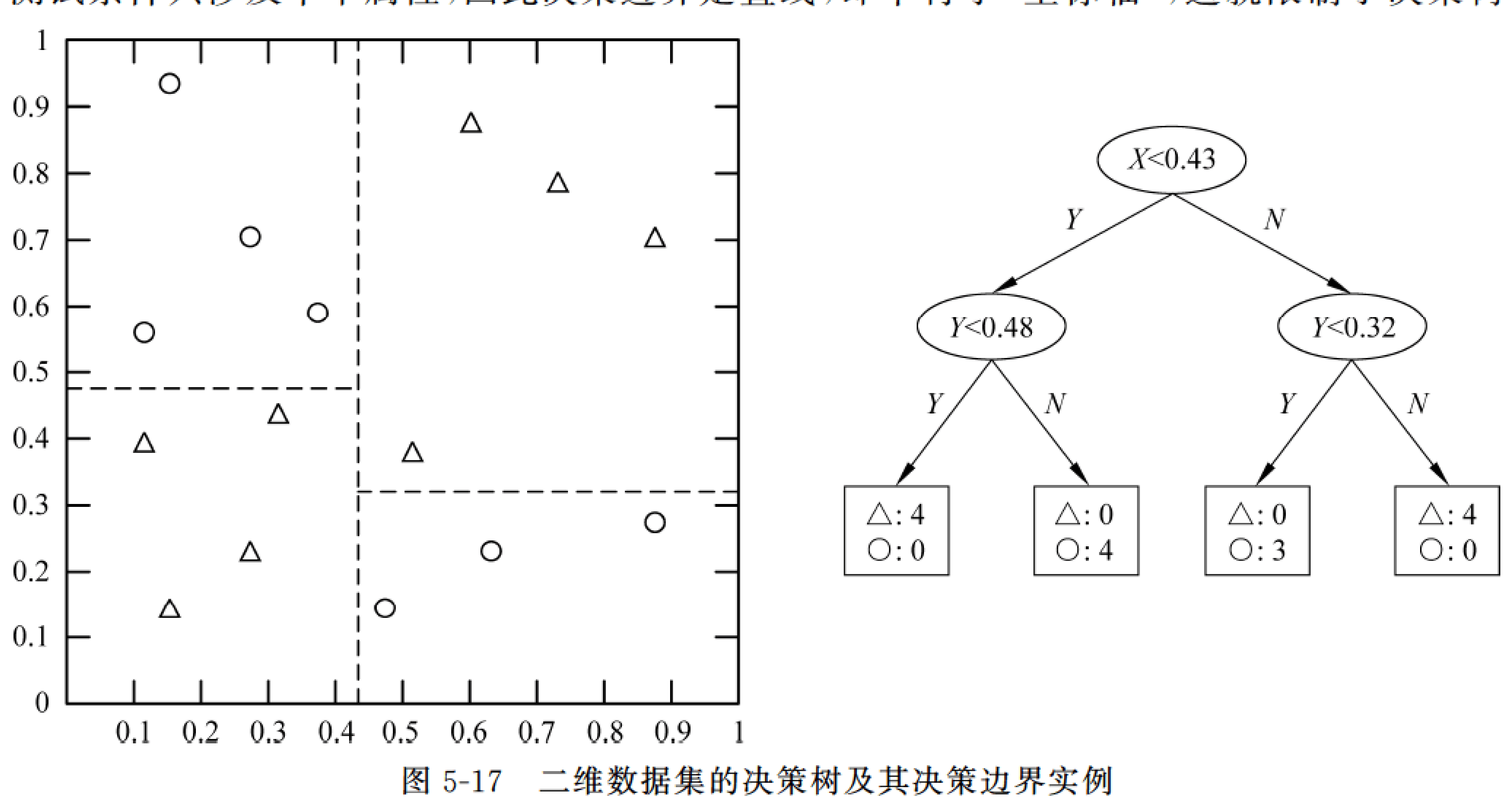


图5-17　二维数据集的决策树及其决策边界实例

对连续属性之间复杂关系建模的表达能力。图 5-18 显示了一个数据集，使用一次只涉及一个属性的测试条件的决策树算法很难有效地对它进行分类。

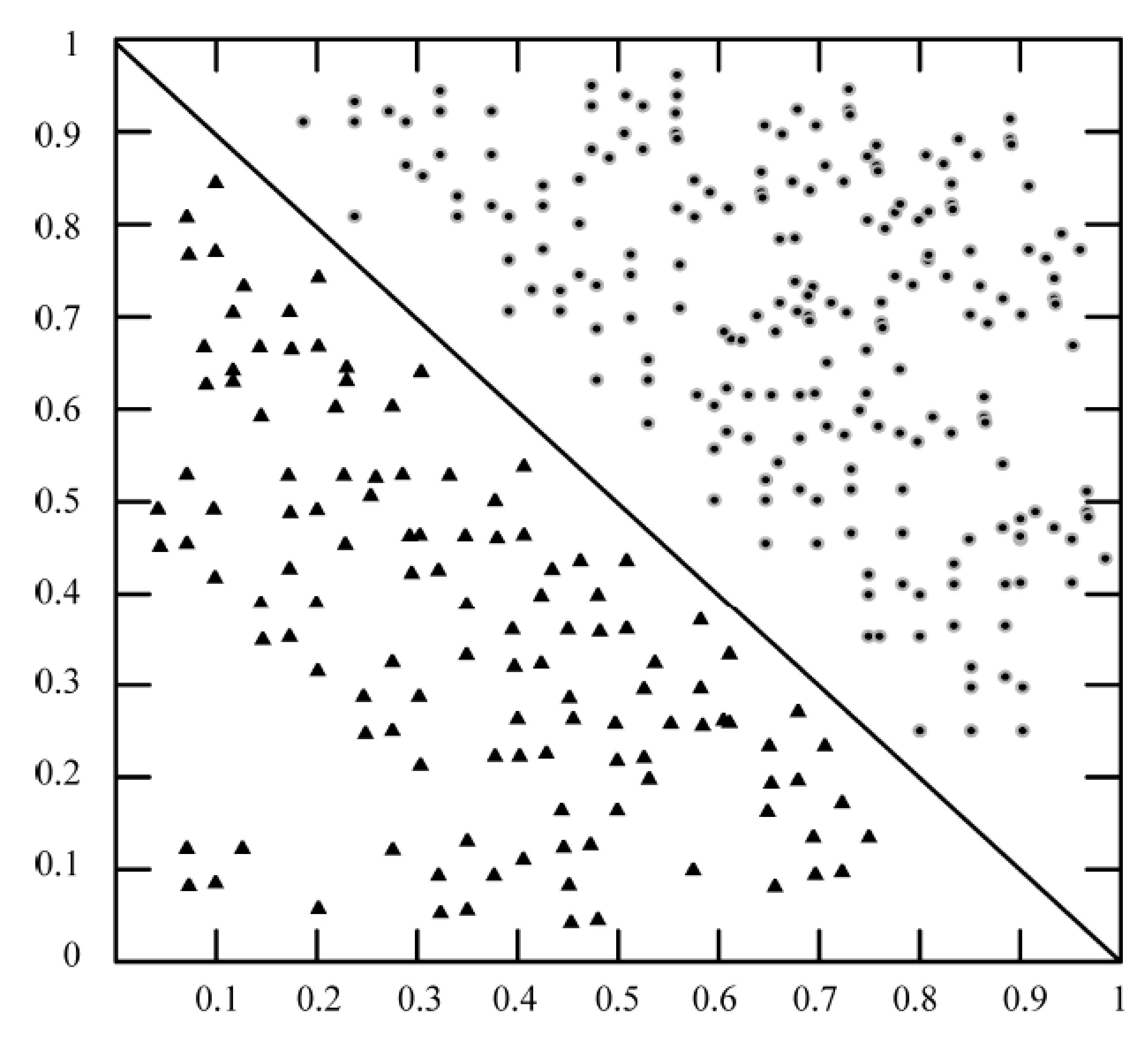


图 5-18　使用仅涉及单个属性的测试条件不能有效划分的数据集的例子

斜决策树(oblique decision tree)可以克服以上的局限，因为它允许测试条件涉及多个属性。图 5-18 中的数据集可以很容易地用斜决策树表示，该斜决策树只有一个基点，其测试条件为

$$x+y<1$$

尽管这种技术具有更强的表达能力，并且能够产生更紧凑的决策树，但是为给定的节点找出最佳测试条件的计算可能是相当复杂的。

归纳构造(constructive induction)提供另一种将数据划分成齐次非矩形区域的方法，该方法创建复合属性，代表已有属性的算术或逻辑组合。新属性提供了更好的类区分能力，并在决策树归纳之前就增广到数据集中。与斜决策树不同，归纳构造不需要昂贵的花费，因为在构造决策树之前，它只需要一次性地确定属性的所有相关组合。相比之下，在扩展每个内部节点时，斜决策树都需要动态地确定正确的属性组合。然而，归纳构造会产生冗余的属性，因为新创建的属性是已有属性的组合。

(11) 研究表明，不纯性度量方法的选择对决策树算法的性能影响很小，这是因为许多度量方法是相互一致的。实际上，树剪枝对最终决策树的影响比不纯性度量的选择的影响更大。

5.3　贝叶斯分类算法

在很多应用中，属性集和类变量之间的关系是不确定的。换句话说，尽管测试记录的属性集和某些训练样例相同，但是也不能正确地预测它的类标号。这种情况产生的原因

可能是噪声,或者出现了某些影响分类的因素却没有包含在分析中。例如,考虑根据一个人的饮食和锻炼的频率来预测他是否有患心脏病的危险。尽管大多数饮食健康、经常锻炼身体的人患心脏病的几率较小,但仍有人由于遗传、过量抽烟、酗酒等其他原因而患病。确定一个人的饮食是否健康、体育锻炼是否充分也是需要论证的课题,这反过来也会给学习问题带来不确定性。

贝叶斯分类法是统计学分类方法,可以预测类隶属关系的概率,如一个给定的元组属于一个特定类的概率。

贝叶斯分类基于贝叶斯定理。分类算法的比较研究发现,一种称为朴素贝叶斯分类法的简单贝叶斯分类法可以与决策树和经过挑选的神经网络分类器相媲美。用于大型数据库时,贝叶斯分类法也表现出高准确率和高速度。

朴素贝叶斯分类法假定一个属性值在给定类上的影响独立于其他属性的值。这一假定称为类条件独立性。做此假定是为了简化计算,并在此意义下称为"朴素的"。

5.3.1 节介绍基本的概率概念和贝叶斯定理。5.3.2 节学习如何进行贝叶斯分类。5.3.3 节则对贝叶斯信念网络做基本介绍。

5.3.1　贝叶斯定理

贝叶斯定理用 Thomas Bayes 的名字命名。Thomas Bayes 是一位不墨守成规的英国牧师,是 18 世纪概率论和决策论的研究者。设 X 是数据元组。在贝叶斯的术语中,X 看做"证据"。通常,X 用 n 个属性集的测量值描述。令 H 为某种假设,如数据元组 X 属于某个特定类 C。对于分类问题,希望确定给定"证据"或观测数据元组 X,假设 H 成立的概率 $P(H|X)$。换言之,给定 X 的属性描述,找出元组 X 属于类 C 的概率。

$P(H|X)$是后验概率(posterior probability),或在条件 X 下 H 的后验概率。例如,假设数据元组世界限于分别由属性 age 和 income 描述的顾客,而 X 是一位 35 岁的顾客,其收入为 4 万美元。令 H 为某种假设,如顾客将购买计算机。则 $P(H|X)$反映了当我们知道顾客的年龄和收入时,顾客 X 将购买计算机的概率。

相反,$P(H)$是先验概率(prior probability),或在条件 X 下 H 的先验概率。对于上面的例子,它是任意给定顾客将购买计算机的概率,而不管他们的年龄、收入或任何其他信息。后验概率 $P(H|X)$比先验概率 $P(H)$基于更多的信息(例如顾客的信息)。$P(H)$独立于 X。

类似地,$P(X|H)$是条件 H 下 X 的后验概率。也就是说,它是已知顾客 X 将购买计算机时,该顾客是 35 岁并且收入为 4 万美元的概率。

$P(X)$是 X 的先验概率。使用上面的例子,它是顾客集合中年龄为 35 岁并且收入为 4 万美元的概率。

如何估计这些概率?正如下面将看到的,$P(X)$、$P(H)$和 $P(X|H)$可以由给定的数据估计。贝叶斯定理的作用是提供了一种由 $P(X)$、$P(H)$和 $P(X|H)$计算后验概率 $P(H|X)$的方法。贝叶斯定理可以用下式表示

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)} \tag{5-7}$$

5.3.2 朴素贝叶斯分类

朴素贝叶斯(Naïve Bayesian)分类法或简单贝叶斯分类法的工作过程如下：

(1) 设 D 是训练元组和它们关联的类标号的集合。通常，每个元组用一个 n 维属性向量 $\boldsymbol{X}=\{x_1,x_2,\cdots,x_n\}$ 表示，描述由 n 个属性 $A_1,A_2,\cdots,A_n$ 对元组的 n 个测量。

(2) 假定有 m 个分类 $C_1,C_2,\cdots,C_m$。给定元组 X，分类法将预测 X 属于具有最高后验概率的类(在条件 X 下)。也就是说，朴素贝叶斯分类法预测 X 属于类 C_i，当且仅当

$$P(C_i \mid X) > P(C_j \mid X) \quad 1 \leqslant j \leqslant m,\ j \neq i$$

这样，最大化 $P(C_i|X)$。$P(C_i|X)$ 最大的类 C_i 称为最大后验假设。根据贝叶斯定理(式(5-7))，有

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)} \tag{5-8}$$

(3) 由于 $P(X)$ 对所有类为常数，所以只需要 $P(X|C_i)P(C_i)$ 最大即可。如果类的先验概率未知，则通常假定这些类是等概率的，即 $P(C_1)=P(C_2)=\cdots=P(C_m)$，并据此对 $P(X|C_i)$ 最大化；否则，最大化 $P(X|C_i)P(C_i)$。注意，类先验概率可以用 $P(C_i)=|C_{i,D}|/|D|$ 估计，其中 $|C_{i,D}|$ 是 D 中类 C_i 的训练元组数。

(4) 给定具有许多属性的数据集，计算 $P(X|C_i)$ 的开销可能非常大。为了降低计算 $P(X|C_i)$ 的开销，可以做**类条件独立**的朴素假定。给定元组的类标号，假定属性值有条件地相互独立(即属性之间不存在依赖关系)。因此，有

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i)P(x_2 \mid C_i)\cdots P(x_n \mid C_i) \tag{5-9}$$

可以很容易地由训练元组估计概率 $P(x_1|C_i),P(x_2|C_i),\cdots,P(x_n|C_i)$。注意，$x_k$ 表示元组 X 在属性 A_k 的值。对于每个属性，考察该属性是分类的还是连续值的。例如，为了计算 $P(X|C_i)$，考虑如下情况：

- 如果 A_k 是分类属性，则 $P(x_k|C_i)$ 是 D 中属性 x_k 的 C_i 类的元组数除以 D 中 C_i 类元组数 $|C_{i,D}|$。
- 如果 A_k 是连续值属性，则需要多做一点工作，但是计算很简单。通常，假定连续值属性服从均值为 μ、标准差为 σ 的高斯分布，由下式定义：

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5-10}$$

因此：

$$P(x_k \mid C_i) = g(x_k,\mu_{C_i},\sigma_{C_i}) \tag{5-11}$$

这些公式看上去可能有点令人生畏，但是沉住气，需要计算 μ_{C_i} 和 σ_{C_i}，它们分别是 C_i 类训练元组属性 A_k 的均值和标准差。将这两个量与 x_k 一起带入式(5-10)，计算 $P(x_k|C_i)$。

例如，设 $X=(35,40000)$，其中 A_1 和 A_2 分别是属性 age 和 income。设类标号属性为 buys_computer。X 相关联的类标号是“yes”(即 buys_computer＝yes)。假设 age 尚未离散化，因此是连续值属性。假设从数据集发现 D 中购买计算机的顾客年龄为 38 ± 12 岁。换言之，对于属性 age 和这个类，有 $\mu=38$ 和 $\sigma=12$。可以把这些量与元组 X 的 $x_1=$

35 一起带入式(5-10),估计 $P(age=35|buys_computer=yes)$。

(5) 为了预测 X 的类标号,对每个类 C_i,计算 $P(X|C_i)P(C_i)$。该分类法预测输入元组 X 的类为 C_i,当且仅当

$$P(X \mid C_i)P(C_i) > P(X \mid C_j)P(C_j), \quad 1 \leqslant j \leqslant m, j \neq i \tag{5-12}$$

换言之,被预测的类标号是使 $P(X|C_i)P(C_i)$ 最大的类 C_i。

该分类法与决策树和神经网络分类法的各种比较试验表明,在某些领域,贝叶斯分类法具有最小的错误率。然而,在实际中并非总是如此。这是由于对其使用的假定(如类条件独立性)的不正确性以及缺乏可用的概率数据造成的。

贝叶斯分类还可以用来为不直接使用贝叶斯定理的其他分类法提供理论判定。例如,在某种假定下,可以证明:与朴素贝叶斯分类法一样,许多神经网络和曲线拟合算法输出最大的后验假定。

考虑图 5-19(a)中的数据集。可以计算每个分类属性的类条件概率,同时利用前面介绍的方法计算连续属性的样本均值和方差。这些概率汇总在图 5-19(b)中。

Tid	有房	婚姻状况	年收入	拖欠贷款
1	是	单身	125k	否
2	否	已婚	100k	否
3	否	单身	70k	否
4	是	已婚	120k	否
5	否	离婚	95k	是
6	否	已婚	60k	否
7	是	离婚	220k	否
8	否	单身	85k	是
9	否	已婚	75k	否
10	否	单身	90k	是

(a) 数据集

P(有房=是|No)=3/7
P(有房=否|No)=4/7
P(有房=是|Yes)=0
P(有房=否|Yes)=1
P(婚姻状况=单身|No)=2/7
P(婚姻状况=离婚|No)=1/7
P(婚姻状况=已婚|No)=4/7
P(婚姻状况=单身|Yes)=2/3
P(婚姻状况=离婚|Yes)=1/3
P(婚姻状况=已婚|Yes)=0
年收入:
- 如果类=No:样本均值=110,样本方差=2975
- 如果类=Yes:样本均值=90,样本方差=25

(b) 概率汇总

图 5-19 贷款分类问题的朴素贝叶斯分类器

为了预测测试记录 X=(有房=否,婚姻状况=已婚,年收入=120k)的类标号,需要计算后验概率 $P(\text{No}|X)$和 $P(\text{Yes}|X)$。

每个类的先验概率可以通过计算属于该类的训练记录所占的比例来估计。因为有 3 个记录属于类 Yes,7 个记录属于类 No,所以 $P(\text{Yes})=0.3$,$P(\text{No})=0.7$。使用图 5-19 中提供的信息,类的条件概率计算如下:

$$\begin{aligned} P(X \mid \text{No}) &= P(\text{有房}=\text{否} \mid \text{No}) \times P(\text{婚姻状况}=\text{已婚} \mid \text{No}) \times P(\text{年收入}=120\text{k} \mid \text{No}) \\ &= 4/7 \times 4/7 \times 0.0072 = 0.0024 \end{aligned}$$

$$\begin{aligned} P(X \mid \text{Yes}) &= P(\text{有房}=\text{否} \mid \text{Yes}) \times P(\text{婚姻状况}=\text{已婚} \mid \text{Yes}) \times P(\text{年收入}=120\text{k} \mid \text{Yes}) \\ &= 1 \times 0 \times 1.2 \times 10^{-9} = 0 \end{aligned}$$

放到一起可得到类 No 的后验概率 $P(\text{No}|X)=\alpha \times 7/10 \times 0.0024=0.0016\alpha$,其中 $\alpha=1/P(X)$,是一个常量。同理,可以得到类 Yes 的后验概率等于 0,这是由于它的类条件概率等于 0。因为 $P(\text{No}|X)>P(\text{Yes}|X)$,所以记录分类为 No。

5.3.3　贝叶斯信念网络

朴素贝叶斯分类假定类条件独立,即给定样本的类标号,属性的值相互条件独立。但在实践中,变量之间的依赖可能存在。贝叶斯信念网络说明联合条件概率分布,它允许在变量的子集间定义类条件独立性。它提供一种因果关系的图形。

举个简单的例子,如图 5-20 所示。对下雨(R)引起草地变湿(W)建模,如图 5-21 所示。天下雨的可能性为 40%,并且下雨时草地变湿的可能性为 90%;也许 10%的时间雨下得不长,不足以让我们真正认为草地被淋湿了。

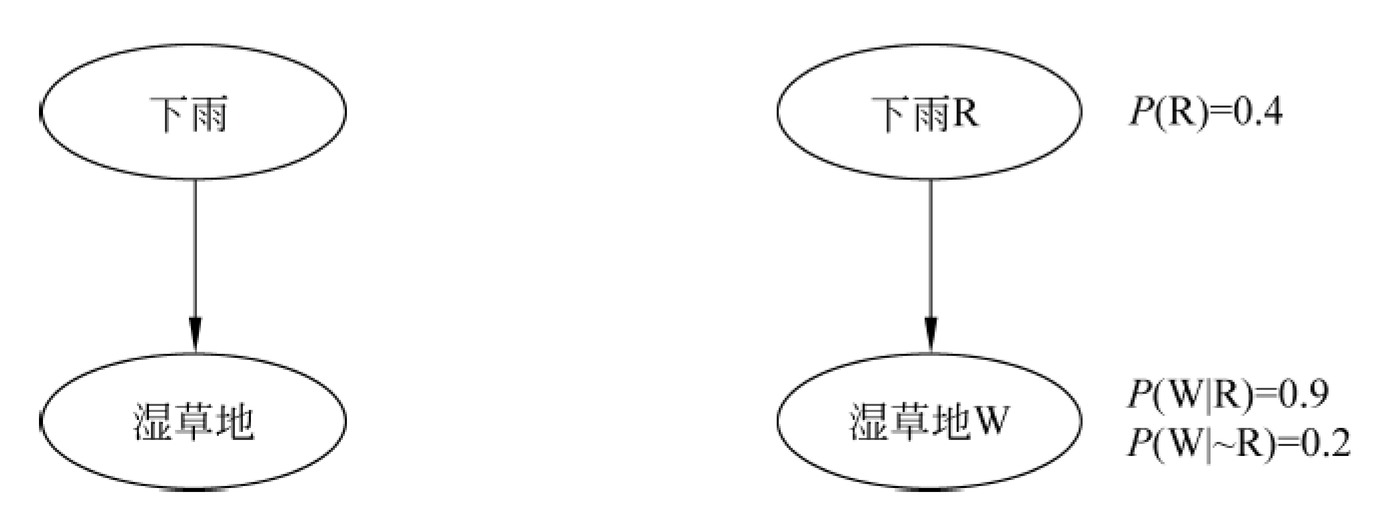


图 5-20　贝叶斯信念网络举例一　　　　图 5-21　对下雨是湿草地的原因建模

在这个例子中,随机变量是二元的:真或假。存在 20%的可能性草地变湿而实际上并没有下雨。例如,使用喷水器时,草地会变湿但并没有下雨。

可以看到 3 个值就可以完全指定 $P(\mathrm{R},\mathrm{W})$ 的联合分布。如果 $P(\mathrm{R})=0.4$,则 $P(\sim\mathrm{R})=0.6$。类似地,$P(\sim\mathrm{W}\mid\mathrm{R})=0.1$,而 $P(\sim\mathrm{W}\mid\sim\mathrm{R})=0.8$。这是一个因果图,解释草地变湿的主要原因是下雨。可以颠倒因果关系并且做出诊断。例如,已知草地是湿的,则下过雨的概率可以计算如下:

$$
\begin{aligned}
P(\mathrm{R}\mid\mathrm{W}) &= \frac{P(\mathrm{W}\mid\mathrm{R})P(\mathrm{R})}{P(\mathrm{W})} \\
&= \frac{P(\mathrm{W}\mid\mathrm{R})P(\mathrm{R})}{P(\mathrm{W}\mid\mathrm{R})P(\mathrm{R})+P(\mathrm{W}\mid\sim\mathrm{R})P(\sim\mathrm{R})} \\
&= \frac{0.9\times0.4}{0.9\times0.4+0.2\times0.6} = 0.75
\end{aligned}
$$

现在,假设想把喷水器(S)作为草地变湿的另一个原因,如图 5-22 所示。对喷水器(S)和下雨(R)使草地变湿(W)建模,如图 5-23 所示。

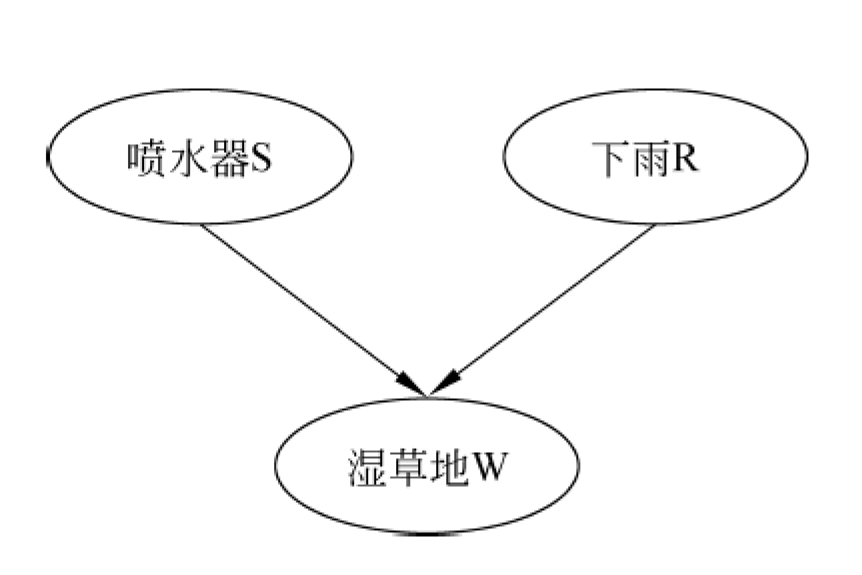


图 5-22　贝叶斯信念网络举例二

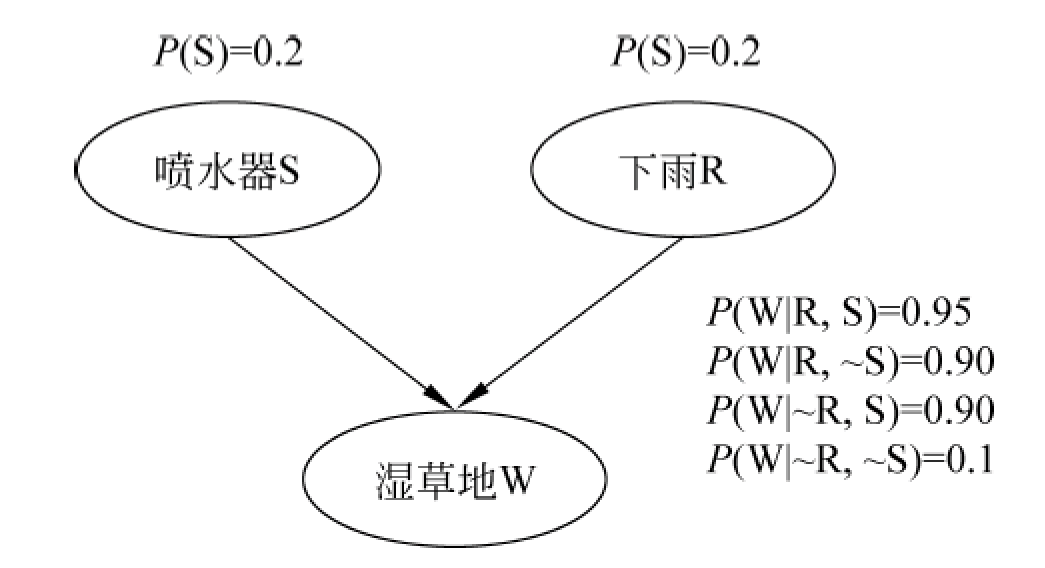


图 5-23　对喷水器和下雨是湿草地的原因建模

节点 W 有两个父节点 R 和 S,因此它的概率是这两个值上的条件概率 $P(W|R,S)$。可以计算喷水器开着草地会湿的概率。这是一个因果(预测)推理:

$$\begin{aligned} P(W \mid S) &= P(W \mid R,S)P(R \mid S) + P(W \mid \sim R,S)P(\sim R \mid S) \\ &= P(W \mid R,S)P(R) + P(W \mid \sim R,S)P(\sim R) \\ &= 0.95 \times 0.4 + 0.9 \times 0.6 = 0.92 \end{aligned}$$

给定草地是湿的,能够计算喷水器开着的概率。这是一个诊断推理:

$$\begin{aligned} P(S \mid W) &= \frac{P(W \mid S)P(S)}{P(W)} \\ &= \frac{0.92 \times 0.2}{0.52} = 0.35 \end{aligned}$$

其中:

$$\begin{aligned} P(W) &= P(W \mid R,S)P(R,S) + P(W \mid \sim R,S)P(\sim R,S) \\ &\quad + P(W \mid R,\sim S)P(R,\sim S) + P(W \mid \sim R,\sim S)P(\sim R,\sim S) \\ &= P(W \mid R,S)P(R)P(S) + P(W \mid \sim R,S)P(\sim R)P(S) \\ &\quad + P(W \mid R,\sim S)P(R)P(\sim S) + P(W \mid \sim R,\sim S)P(\sim R)P(\sim S) \\ &= 0.95 \times 0.4 \times 0.2 + 0.9 \times 0.6 \times 0.2 + 0.9 \times 0.4 \times 0.8 + 0.1 \times 0.6 \times 0.8 \\ &= 0.52 \end{aligned}$$

知道草是湿的增加了喷水器开着的可能。现在假设下过雨,有

$$P(S \mid R,W) = \frac{P(W \mid R,S)P(S \mid R)}{P(W \mid R)} = \frac{P(W \mid R,S)P(S)}{P(W \mid R)} = 0.21$$

注意,这个值比 $P(S|W)$小。这叫做**解释远离**(explaining away)。给定已知下过雨,则喷水器导致湿草地的可能性降低了。已知草地是湿的,下雨和喷水器成为相互依赖的。

5.4 支持向量机算法

支持向量机(Support Vector Machine,SVM)已经成为一种备受关注的分类技术。这种技术具有坚实的统计学理论基础,并在许多实际应用(如手写数字的识别、文本分类等)中展示了大有可为的实践效用。此外,SVM 可以很好地应用于高维数据,避免了高维灾难问题。这种方法具有一个独特的特点,它使用训练实例的一个子集来表示决策边界,该子集称作支持向量(support vector)。

支持向量机的第一篇论文由 Vladimir Vapnik 和他的同事 Bernhard Boser 及 Isabelle Guyon 于 1992 年发表,而其基础工作早在 20 世纪 60 年代就已经出现。尽管最快的 SVM 的训练也非常慢,但是其对复杂的非线性边界的建模是非常准确的。与其他模型相比,SVM 不太容易过分拟合。SVM 还提供了学习模型的紧凑表示。SVM 可以用于数值预测和分类。

5.4.1 节和 5.4.2 节分别就数据线性可分和数据非线性可分的情况作介绍。

5.4.1 数据线性可分的情况

为了解释 SVM,首先考察最简单的情况——二元分类问题,其中两个类是线性可分

的。设给定的数据集 D 为 $(X_1, y_1), (X_2, y_2), \cdots, (X_{|D|}, y_{|D|})$，其中 X_i 是训练元组，具有类标号 y_i。每个 y_i 可以取值 +1 或 −1（即 $y_i \in \{+1, -1\}$），分别对应于类 buys_computer＝yes 和 buys_computer＝no。为了便于可视化，考虑一个基于两个输入属性 A_1 和 A_2 的例子，如图 5-24 所示。从该图可以看出，该二维数据是线性可分的（或简称“线性的”），因为可以画一条直线，把类+1 的元组与类−1 的元组分开。

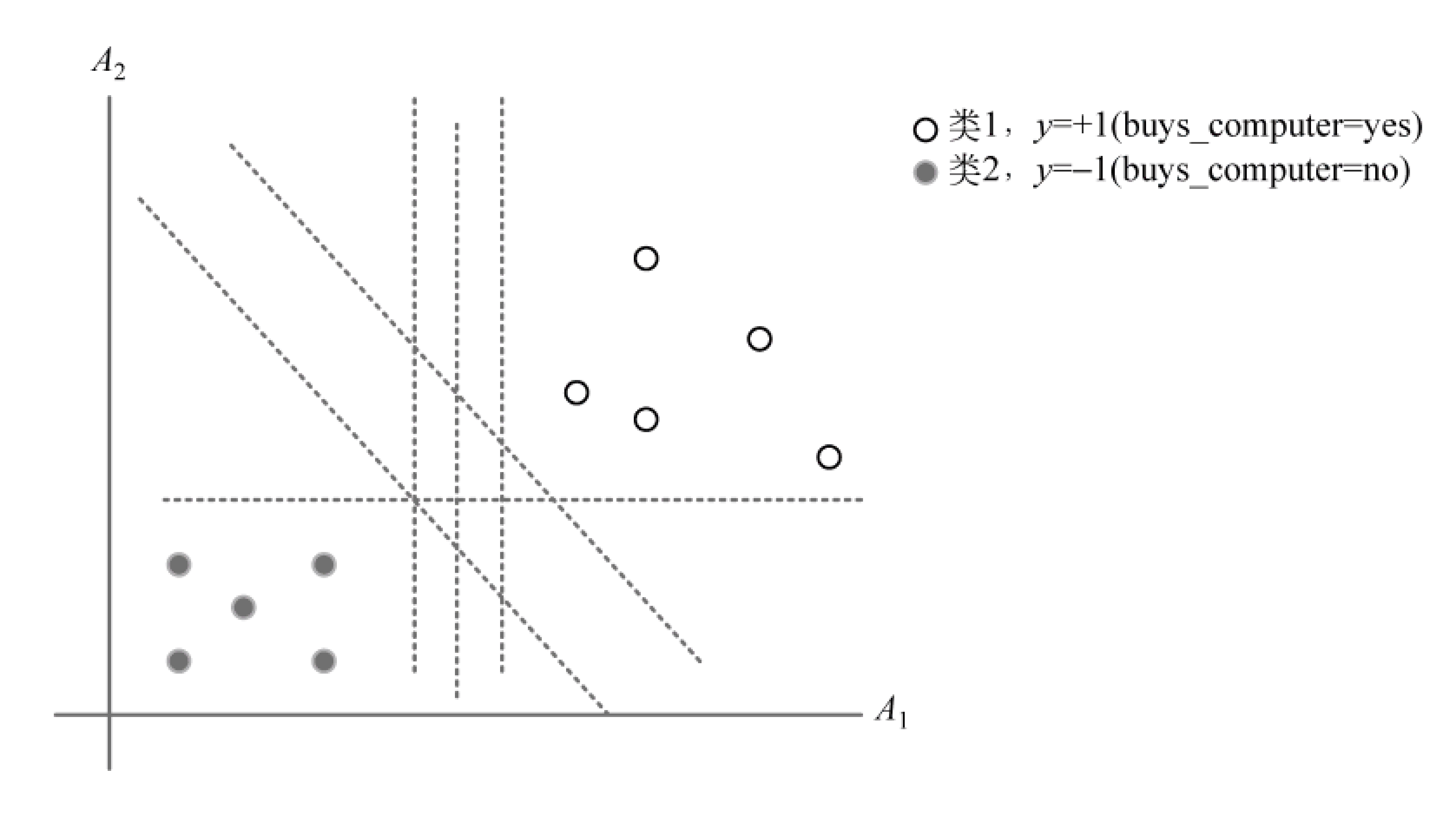


图 5-24　线性可分的 2D 数据集。有无限多个（可能的）分离超平面或决策边界，其中一些用虚线显示

可以画出无限多条分离直线。我们想找出“最好的”一条，即在先前未见到的元组上具有最小分类误差的那一条。如何找到这条最好的直线？注意，如果数据是 3D 的（即具有 3 个属性），则我们希望找出最佳分离平面。推广到 n 维，我们希望找出最佳超平面。下面使用术语“超平面”表示寻找的决策边界，而不管输入属性的个数是多少。这样，换一句话说，如何找出最佳超平面？

SVM 通过搜索**最大边缘超平面**（Maximum Marginal Hyperplane，MMH）来处理该问题。考虑图 5-25，它显示了两个可能的分离超平面和它们的相关联的边缘。在给出边缘的定义之前，让我们先直观地考察该图。两个超平面都对所有的数据元组正确地进行了分类。然而，直观地看，我们预料具有较大边缘的超平面在对未来的数据元组分类上比具有较小边缘的超平面更准确。这就是为什么（在学习或训练阶段）SVM 要搜索具有最大边缘的超平面，即最大边缘超平面。MMH 相关联的边缘给出类之间的最大分离性。

关于边缘的非形式化定义，可以说从超平面到其边缘的一个侧面的最短距离等于从该超平面到其边缘的另一个侧面的最短距离，其中边缘的“侧面”平行于超平面。事实上，在处理 MMH 时，这个距离是从 MMH 到两个类的最近的训练元组的最短距离。

分离超平面可以记为

$$W \cdot X + b = 0 \tag{5-13}$$

其中，W 是权重向量，即 $W = \{w_1, w_2, \cdots, w_n\}$；$n$ 是属性数；b 是标量，通常称作偏倚（bias）。为了便于观察，考虑两个输入属性 A_1 和 A_2，如图 5-25(b)所示。训练元组是二维的，如 $X = (x_1, x_2)$，其中 x_1 和 x_2 分别是 A_1 和 A_2 上的值。如果把 b 看做附加的权重

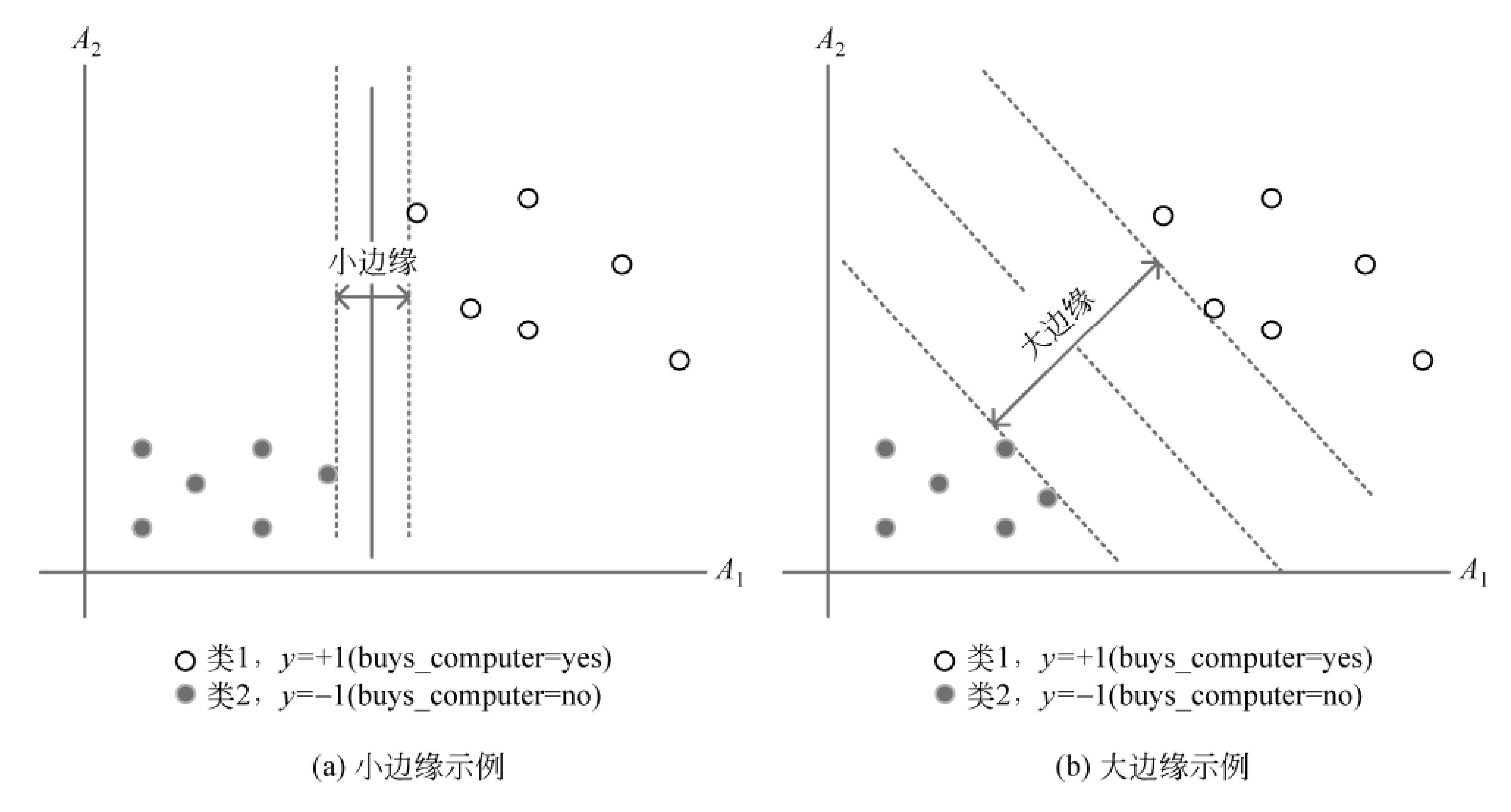


图 5-25　两个可能的分离超平面和它们的边缘

w_0，则可以把分离超平面改写成

$$w_o + w_1 x_1 + w_2 x_2 = 0 \tag{5-14}$$

这样，位于分离超平面上方的点满足

$$w_o + w_1 x_1 + w_2 x_2 > 0 \tag{5-15}$$

类似地，位于分离超平面下方的点满足

$$w_o + w_1 x_1 + w_2 x_2 < 0 \tag{5-16}$$

可以调整权重使得定义边缘"侧面"的超平面可以记为

$$H_1: w_o + w_1 x_1 + w_2 x_2 \geqslant 1, \quad 对于\ y_i = +1 \tag{5-17}$$

$$H_2: w_o + w_1 x_1 + w_2 x_2 \leqslant -1, \quad 对于\ y_i = -1 \tag{5-18}$$

也就是说，落在 H_1 上或上方的元组都属于类＋1，而落在 H_2 上或下方的元组都属于类－1。结合式(5-17)和式(5-18)这两个不等式，可以得到：

$$y_i(w_0 + w_1 x_1 + w_2 x_2) \geqslant 1, \quad \forall i \tag{5-19}$$

落在超平面 H_1 或 H_2（即定义边缘的"侧面"）上的对于任意训练元组都使式(5-19)的等号成立，称为**支持向量**(support vector)。也就是说，它们离 MMH 一样近。在图 5-26 中，支持向量用加粗的圆圈显示。本质上，支持向量是最难分类的元组，并且给出了最多的分类信息。

由上，可以得到最大边缘的计算公式。从分离超平面到 H_1 上任意点的距离是 $\frac{1}{||W||}$，其中 $||W||$ 是欧几里得范数，即 $\sqrt{W \cdot W}$（如果 $W=\{w_1, w_2, \cdots, w_n\}$，则 $\sqrt{W \cdot W} = \sqrt{W_1^2 + W_2^2 + \cdots + W_n^2}$）。根据定义，它等于 H_2 上任意点到分离超平面的距离。因此，最大边缘是 $\frac{2}{||W||}$。

SVM 如何找出 MMH 和支持向量？使用某种"特殊的数学技巧"，可以改写

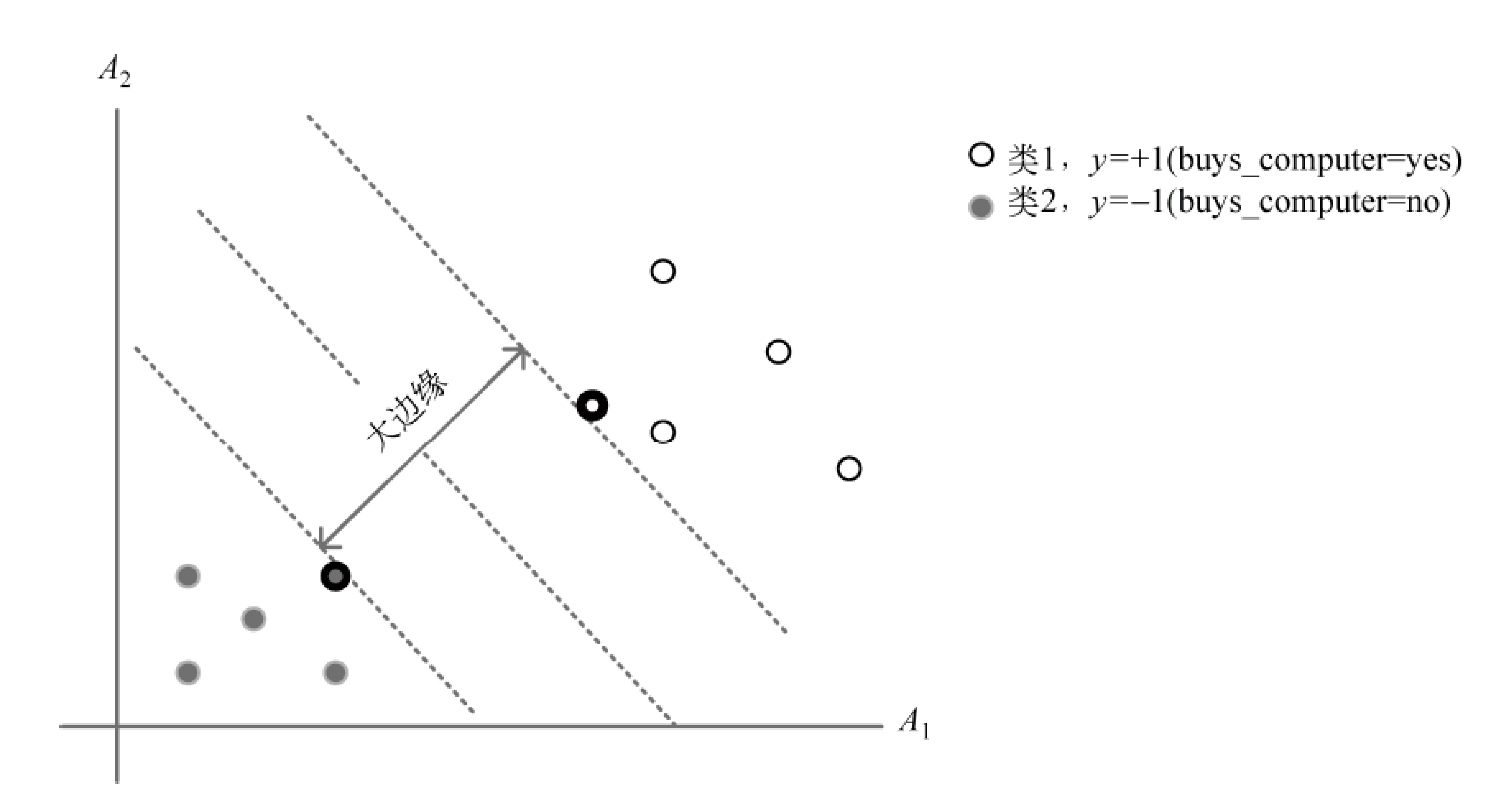


图 5-26　支持向量。SVM 发现最大分离超平面，即与最近的训练元组具有最大距离的超平面

式(5-19)，将它变换成一个被约束的(凸)二次最优化问题。

如果数据很少(例如，少于 2000 个训练元组)，则可以使用任何求解约束的凸二次最优化问题的最优化软件包来找出支持向量和 MMH。对于大型数据，可以使用特殊的、更有效的训练 SVM 的算法。一旦找出支持向量和 MMH(注意，支持向量定义 MMH)，就有了一个训练后的支持向量机。MMH 是一个线性类边界，因此对应的 SVM 可以用来对线性可分数据进行分类。这种训练后的 SVM 称为线性 SVM。

一旦得到训练后的支持向量机，如何用它对检验元组(即新元组)分类？根据拉格朗日公式，最大边缘超平面可以改写成决策边界：

$$d(X^{\mathrm{T}}) = \sum_{i=1}^{l} y_i \alpha_i X_i X^{\mathrm{T}} + b_0 \tag{5-20}$$

其中，y_i 是支持向量 X_i 的类标号，X^{T}是检验元组，α_i 和 b_0是由上面的最优化或 SVM 算法自动确定的数值参数，而 l 是支持向量的个数。

给定检验元组 X^{T}，将它代入式(5-20)，然后检查结果的符号。这将告诉我们检验元组落在超平面的哪一侧。如果该符号为正，则 X^{T}落在 MMH 上或上方，因而 SVM 预测 X^{T}属于类+1(在此情况下，代表 buys_computer＝yes)。如果该符号为负，则 X^{T}落在 MMH 上或下方，因而 SVM 预测 X^{T}属于类−1(代表 buys_computer＝no)。

注意，式(5-20)包含支持向量 X_i 和检验元组 X^T的点积。正如下面要介绍的，当给定数据非线性可分时，这对于发现 MMH 和支持向量是非常有用的。

在考虑非线性可分的情况之前，还有两件重要的事情要注意。学习后的分类器的复杂度由支持向量数而不是由数据的维数刻画。因此，与其他方法相比，SVM 不太容易过分拟合。支持向量是基本或临界的训练元组——它们距离决策边界(MMH)最近。如果删除其他元组并重新训练，则将发现相同的分离超平面。此外，找到的支持向量数可以用来计算 SVM 分类器的期望误差率的上界，这独立于数据的维度。具有少量支持向量的 SVM 可以具有很好的泛化性能，即使数据的维度很高时也是如此。

5.4.2 数据非线性可分的情况

在5.4.1节，我们学习了对线性可分数据分类的线性SVM。但是，如果数据不是线性可分的，如图5-27中的数据，怎么办？在这种情况下，不可能找到一条将这些类分开的直线。我们在上节研究的线性SVM不可能找到可行解，怎么办？

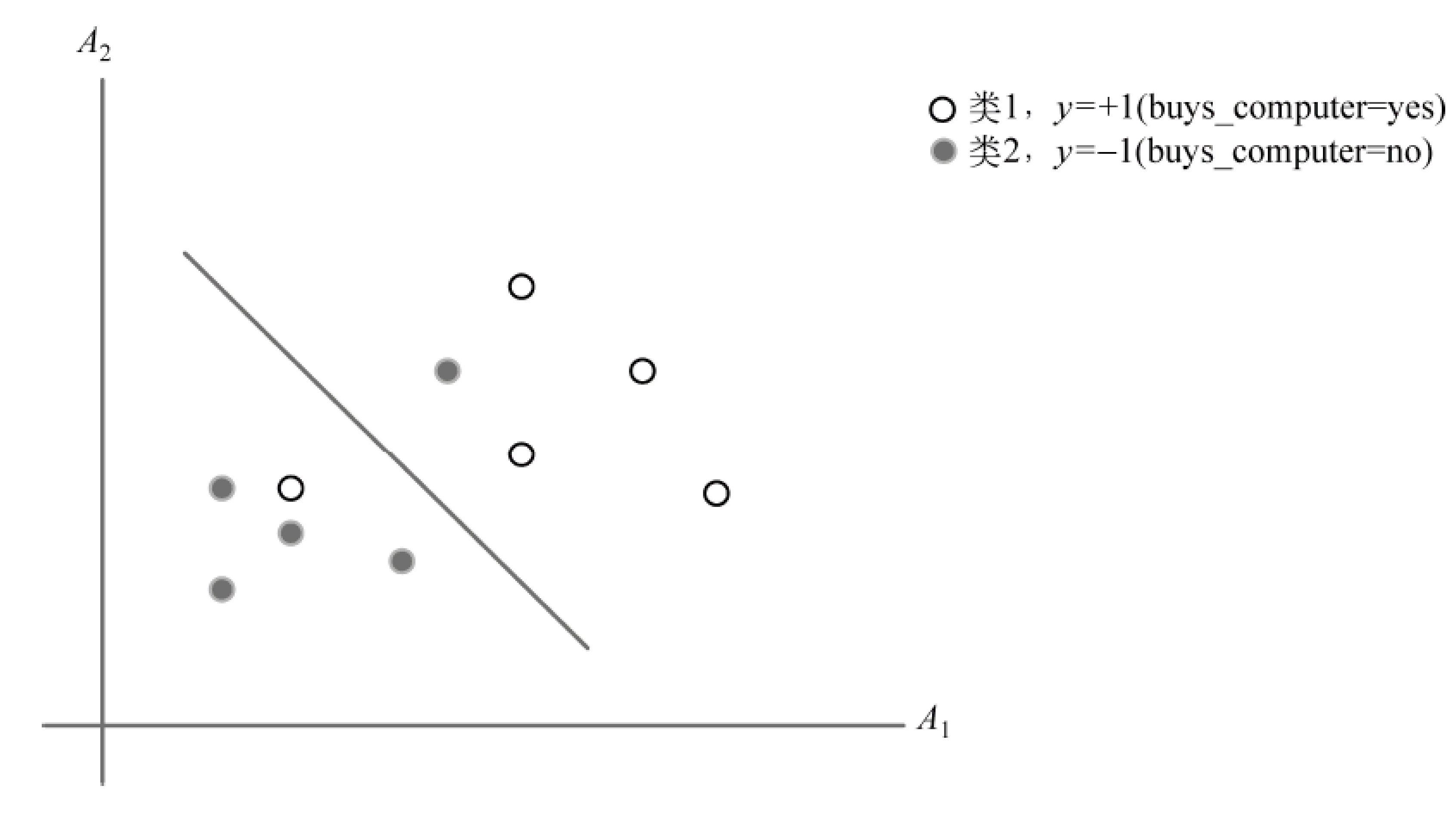


图5-27 线性不可分数据的一个简单二维例子

好消息是，可以扩展上面介绍的线性SVM，为线性不可分的数据（也称非线性可分的数据，或简称非线性数据）的分类创建非线性的SVM。这种SVM能够发现输入空间中的非线性决策边界（即非线性超曲面）。

按如下方法扩展线性SVM的方法，得到非线性的SVM。有两个主要步骤。第一步，用非线性映射把原输入数据变换到较高维空间。这一步可以使用多种常用的非线性映射。一旦将数据变换到较高维空间，第二步就在新的空间搜索分离超平面。此时又遇到二次优化问题，可以用线性SVM公式求解。在新空间找到的最大边缘超平面对应于原空间的非线性分离超曲面。

在求解线性SVM的二次最优化问题时（即在新的较高维空间搜索线性SVM时），训练元组仅出现在形如$\phi(X_i)\cdot\phi(X_j)$的点积中，其中$\phi(X)$是用于训练元组变换的非线性映射函数。结果表明，它完全等价于将核函数$K(X_i,X_j)$应用于原输入数据，而不必在变换后的数据元组上计算点积，即

$$K(X_i,X_j)=\phi(X_i)\cdot\phi(X_j) \tag{5-21}$$

换言之，每当$\phi(X_i)\cdot\phi(X_j)$出现在训练算法中时，都可以用$K(X_i,X_j)$替换它。这样，所有的计算都在原来的输入空间上进行，这可能会降低难度。我们可以避免这种映射，事实上，我们甚至不必知道该映射是什么。使用这种技巧之后，可以找出最大分离超平面。

可以用来替换上面的点积的核函数包括以下几个：

（1）***h*次多项式核函数**：

$$K(X_i,X_j)=(X_i\cdot X_j+1)^h$$

（2）**高斯径向基函数核函数：**

$$K(X_i,X_j)=e^{-||X_i-X_j||^2/2\sigma^2}$$

（3）**S 型核函数：**

$$K(X_i,X_j)=\tanh(\kappa X_i\cdot X_j-\delta)$$

不同的核函数导致(原)输入空间上出现了不同的非线性分类器。熟悉神经网络的读者可能注意到,非线性的 SVM 所发现的决策超曲面与其他著名的神经网络分类器所发现的同属一种类型。例如,具有高斯径向基函数(RBF)的 SVM 与称作径向基函数网络的一类神经网络产生相同的决策超曲面。具有 S 型核的 SVM 等价于一种称作多层感知器(无隐藏层)的简单 2 层神经网络。

没有一种“黄金规则”可以确定哪种可用的核函数将推导出最准确的 SVM。在实践中,核函数的选择一般并不导致结果准确率的很大差别。SVM 训练总是发现全局解,而不像后向传播等神经网络那样常常存在局部最小。

至此已经介绍了二元分类的线性和非线性 SVM。对于多类问题,可以用组合 SVM 分类器。

关于 SVM,主要研究目标是提高训练和检验速度,使得 SVM 可以成为超大型数据集(例如数以百万计的支持向量)更可行的选择。其他问题包括为给定的数据集确定最佳核函数,为多类问题找出更有效的方法。

5.5 粗糙集分类算法

粗糙集理论是 1982 年由波兰的著名科学家 Z. Pawlak 提出的。它是一种能够有效地处理不精确、不确定性数据的数学工具,并且它还具有不需要任何的先验知识,只依赖于数据集本身等优点。粗糙集理论已成为数据挖掘、机器学习等领域的研究热点之一。本节介绍粗糙集分类的相关算法。

粗糙集被广泛应用于数据挖掘与知识发现等领域。为了能够很好地刻画不确定性问题,粗糙集理论主要是根据某个条件属性集所确定的不可分辨关系,并把它称作概念,从而可准确地判断出某样本是否属于该概念,也有可能不能够判断出某样本是否属于该概念。粗糙集采用上下近似集的概念来描述这一问题。其中,下近似集表示可确定能够归入某一感兴趣概念的对象集合,上近似集表示可能属于某一感兴趣概念的对象集合。通过上下近似集可以很好地刻画出数据集的不确定性区域,即边界区域,如图 5-28 所示。由于它能够很好地刻画边界区域,使得粗糙集理论在数据挖掘与知识发现等领域中具有很强的生命力。

粗糙集理论可以用于分类,发现不准确数据或噪声数据内的结构联系。它只能应用于离散数值属性。因此,连续值属性必须在使用前离散化。

粗糙集也可以用于属性子集选择(或特征归约,可以识别和删除无助于给定训练数据分类的属性)和相关分析(根据分类任务评估每个属性的贡献或显著性)。找出可以描述给定数据集中所有概念的最小属性子集(归约集)是 NP 困难的。然而,目前已提出了一些降低计算强度的算法。例如,有一种方法使用识别矩阵(discernibility matrix)存放每

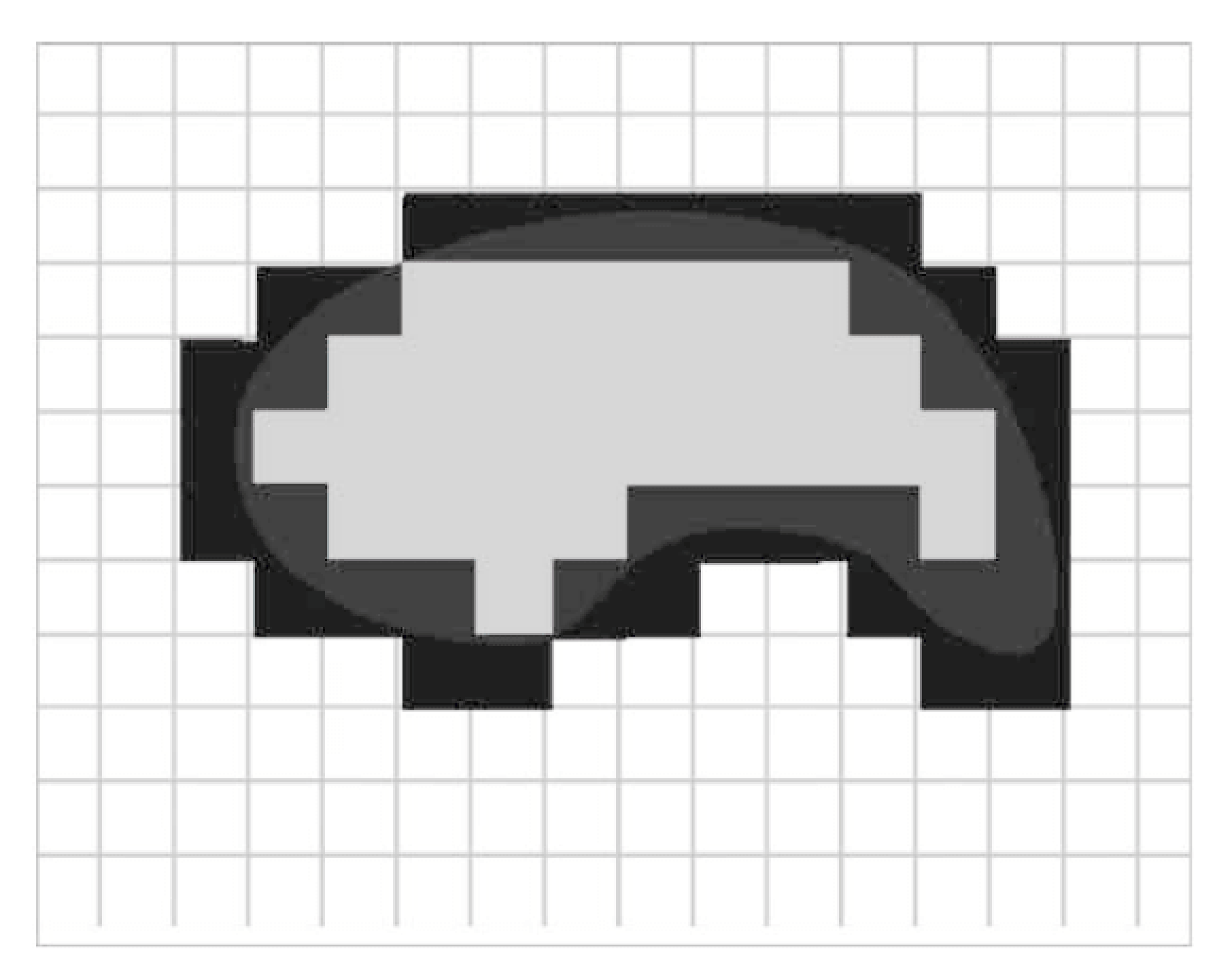


图 5-28 集合 X 的上下近似集和边界区域的图示

对数据元组属性值之差。该方法不是在整个训练集上搜索，而是搜索矩阵，检测冗余属性。

近年来，众多研究学者将其与其他分类算法相结合，取得了很好的成果。其主要表现在两个方面：一方面，粗糙集仅仅计算出信息系统中不精确的区域，对该区域并没有进一步的处理方法，因此研究学者们将其他分类算法应用于粗糙集边界区域，从而能够很好地与粗糙集结合起来，提高了其分类决策能力；另一方面，粗糙集属性约简过程能够删除信息系统不必要的属性，从而简化信息系统。在多维数据分析中，为了删除冗余属性，降低噪声的影响，通常将粗糙集属性约简应用于其他分类算法中，作为其他分类算法的预处理过程，从而提高其他分类算法的正确率。

5.6 分类器评估方法

既然已经建立了分类模型，你的脑海中就可能浮现许多问题。例如，假设使用先前的销售数据训练分类器，预测顾客的购物行为。你希望评估该分类器预测未来顾客购物行为(即未经过训练的未来顾客数据)的准确率。你甚至可能尝试用不同的方法建立多个分类器，并且希望比较它们的准确率。但是，什么是准确率？如何估计它？分类器准确率的某些度量比其他度量更合适吗？如何得到可靠的准确率估计？

5.6.1 节介绍分类器准确率的各种评估度量。保持和随机子抽样(5.6.2 节)、k 折交叉验证(5.6.3 节)和自助方法(5.6.4 节)都是基于给定数据的随机抽样划分。5.6.5 节讨论如何使用统计显著性检验来评估两个分类器的准确率之差是否纯属偶然。

5.6.1 评估分类器性能的度量

本节介绍一些评估度量，用来评估分类器预测元组类标号的性能或准确率。这里既考虑各类元组大致均匀分布的情况，也考虑类不平衡的情况(例如，在医学化验中，感兴趣的重要类稀少)。本节介绍的分类器评估度量汇总在表 5-4 中，包括准确率(又称为识别率)、敏感度(或称为召回率)、特效性、精度、F_1 和 F_β。表中，某些度量有多个名称。TP、TN、FP、FN、P、N 分别代表真正例、真负例、假正例、假负例、正和负样本数。注意，尽管准确率是一个特定的度量，但是“准确率”一词也是经常用于谈论分类器预测能力的通用术语。

表 5-4　分类器、评估度量

度　　量	公　　式
准确率、识别率	$\frac{TP+TN}{P+N}$
错误率、误分类率	$\frac{FP+FN}{P+N}$
敏感度、真正例率、召回率	$\frac{TP}{P}$
特效性、真负例率	$\frac{TN}{N}$
精度	$\frac{TP}{TP+FP}$
F、F_1、F 分数 精度和召回率的调和均值	$\frac{2\times precision\times recall}{precision+recall}$
F_β，其中 β 是非负实数	$\frac{(1+\beta^2)\times precision\times recall}{\beta^2\times precision+recall}$

还有 4 个需要知道的术语。这些术语是用于计算许多评估度量的“构件”，理解它们有助于领会各种度量的含义。

- 真正例/真阳性(True Positive，TP)：是指被分类器正确分类的正元组。令 TP 为真正例的个数。
- 真负例/真阴性(True Negative，TN)：是指被分类器正确分类的负元组。令 TN 为真负例的个数。
- 假正例/假阳性(False Positive，FP)：是被错误地标记为正元组的负元组(例如，类 buys_computer＝no 的元组被分类器预测为 buys_computer＝yes)。令 FP 为假正例的个数。
- 假负例/假阴性(False Negative，FN)：是被错误地标记为负元组的正元组(例如，类 buys_computer＝yes 的元组被分类器预测为 buys_computer＝no)。令 FN 为假负例的个数。

这些术语汇总在图 5-29 的混淆矩阵(confusion matrix)中。

混淆矩阵是分析分类器识别不同类元组的一种有用工具。TP 和 TN 告诉我们分类

		预测的类		合计
		yes	no	
实际的类	yes	TP	FN	P
	no	FP	TN	N
	合计	P'	N'	$P+N$

图 5-29　一个混淆矩阵，显示了正元组和负元组的合计

器何时分类正确，而 FP 和 FN 告诉我们分类器何时分类错误。给定 m 个类（$m \geqslant 2$），**混淆矩阵**是一个至少为 $m \times m$ 的表。前 m 行和 m 列中表项 $CM_{i,j}$ 表示类 i 的元组被分类器标记为类 j 的个数。理想地，对于具有高准确率的分类器，大部分元组应该被混淆矩阵从 $CM_{1,1}$ 到 $CM_{m,m}$ 的对角线上的表项表示，而其他表项为 0 或者接近 0。也就是说，FP 和 FN 接近 0。

该表可能有附加的行和列，提供合计。例如，在图 5-29 的混淆矩阵中，显示了 P 和 N。此外，P' 开始被分类器标记为正的元组数（TP＋FP），N' 被标记为负的元组数（TN＋FN）。元组的总数为 TP＋TN＋FP＋PN，或 $P+N$，或 $P'+N'$。注意，尽管所显示的混淆矩阵是针对二元分类问题的，但是很容易用类似的方法给出多类问题的混淆矩阵。

现在，从准确率开始考察评估度量。分类器在给定检验集上的**准确率**（accuracy）是被该分类器正确分类的元组所占的百分比，即

$$\text{accuracy} = \frac{TP + TN}{P + N} \tag{5-22}$$

在模式识别文献中，准确率又被称为分类器的总体**识别率**，即它反映分类器对各类元组的正确识别情况。

也可以评估分类器 M 的错误率或误分类率，它是 $1-\text{accuracy}(M)$，其中 $\text{accuracy}(M)$ 是 M 的准确率。它也可以用下式计算

$$\text{error_rate} = \frac{FP + FN}{P + N} \tag{5-23}$$

如果想使用训练集（而不是检验集）来估计模型的错误率，则该量称为**再代入误差**（resubstitution error）。这种错误估计是实际错误率的乐观估计（类似地，对应的准确率估计也是乐观的），因为并未在没有见过的样本上对模型进行检验。

现在，考虑类不平衡问题，其中感兴趣的主类是稀少的。也就是说，数据集的分布反映负类显著地占多数，而正类占少数。例如，在欺诈检测应用中，感兴趣的类（或正类）是 fraud（欺诈），它的出现远不及负类 nonfraudulant（非欺诈）频繁。在医疗数据中，可能也有稀有类，如 cancer，而可能的类值是 yes 和 no。97％的准确率使得分类器看上去相当准确，但是，如果实际只有 3％的训练元组是癌症，显然，97％的准确率可能不是可接受的。例如，该分类器可能只是正确地标记了非癌症元组，而对所有癌症元组的分类都是错误的。因此，需要其他的度量来评估分类器正确地识别正元组（cancer＝yes）的情况和正确地识别负元组（cancer＝no）的情况。

为此，可以分别使用**灵敏性**（sensitivity）和**特效性**（specificity）度量。灵敏度也称为真正例（识别）率（即正确识别的正元组的百分比），而特效性是真负例率（即正确识别的负

元组的百分比)。这两个度量定义为

$$\text{sensitivity} = \frac{\text{TP}}{P} \tag{5-24}$$

$$\text{specificity} = \frac{\text{TN}}{N} \tag{5-25}$$

可以证明准确率是灵敏性和特效性度量的函数:

$$\text{accuracy} = \text{sensitivity}\frac{P}{P+N} + \text{specificity}\frac{N}{P+N} \tag{5-26}$$

精度和召回率度量也在分类中广泛使用。**精度**(precision)可以看作是精确性的度量(即标记为正类的元组中实际为正类所占的百分比),而**召回率**(recall)是完全性的度量(即正元组标记为正的百分比)。召回率看上去熟悉,因为它就是灵敏性(或真正例率)。这些度量可以用下面的公式计算:

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{5-27}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{\text{TP}}{P} \tag{5-28}$$

另一种使用精度和召回率的方法是把它们组合到一个度量中。这是 F 度量(又称为 F_1 分数或 F 分数)和 F_β 度量的方法,定义如下:

$$\text{F} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5-29}$$

$$F_\beta = \frac{(1+\beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}} \tag{5-30}$$

其中,β 是非负实数。F 度量是精度召回率的调和均值。它赋予精度和召回率相等的权重。F_β 度量是精度和召回率的加权度量。它赋予召回率的权重是赋予精度的权重的 β 倍。通常使用的 F_β 是 F_2(它赋予召回率的权重是精度的 2 倍)和 $F_{0.5}$(它赋予精度的权重是召回率的 2 倍)。

在分类问题中,通常假定所有的元组都是唯一可分类的,即每个训练元组都只属于一个类。然而,由于大型数据库中的数据非常多样化,因此假定每个元组可以属于多个类是更可行的。这样,如何度量大型数据库上分类器的准确率呢?准确率度量是不合适的,因为它没考虑元组属于多个类的可能性。

有用的是返回类分布概率,而不是返回类标号。这样,准确率度量可以采用**二次猜测**(second guess)试探:如果一个类预测与最可能的或次可能的类一致,则这个类预测被认为是正确的。尽管这在某种程度上确实考虑了元组的非唯一分类,但它不是完全解。

除了基于准确率的度量外,还可以根据其他比较分类器:

- **速度**。这涉及产生和使用分类器的计算开销。
- **鲁棒性**。这是假定数据有噪声或有缺失值时分类器做出正确预测的能力。通常,鲁棒性用噪声和缺失值渐增的一系列合成数据集评估。
- **可伸缩性**。这涉及给定大量数据,有效地构造分类器的能力。通常,可伸缩性用规模渐增的一系列数据评估。
- **可解释性**。这涉及分类器或预测器提供的理解和洞察水平。可解释性是主观的,

因而很难评估。决策树和分类规则可能容易解释，但随着它们变得复杂，它们的可解释性也随之消失。

5.6.2 保持方法和随机二次抽样

保持(holdout)方法是迄今为止讨论准确率时暗指的方法。在这种方法中，给定数据随机地划分成两个独立的集合：训练集和检验集。通常，2/3 的数据分配到训练集，其余 1/3 分配到检验集。使用训练集导出模型，其准确率用检验集估计(见图 5-30)。估计是悲观的，因为只有一部分初始数据用于导出模型。

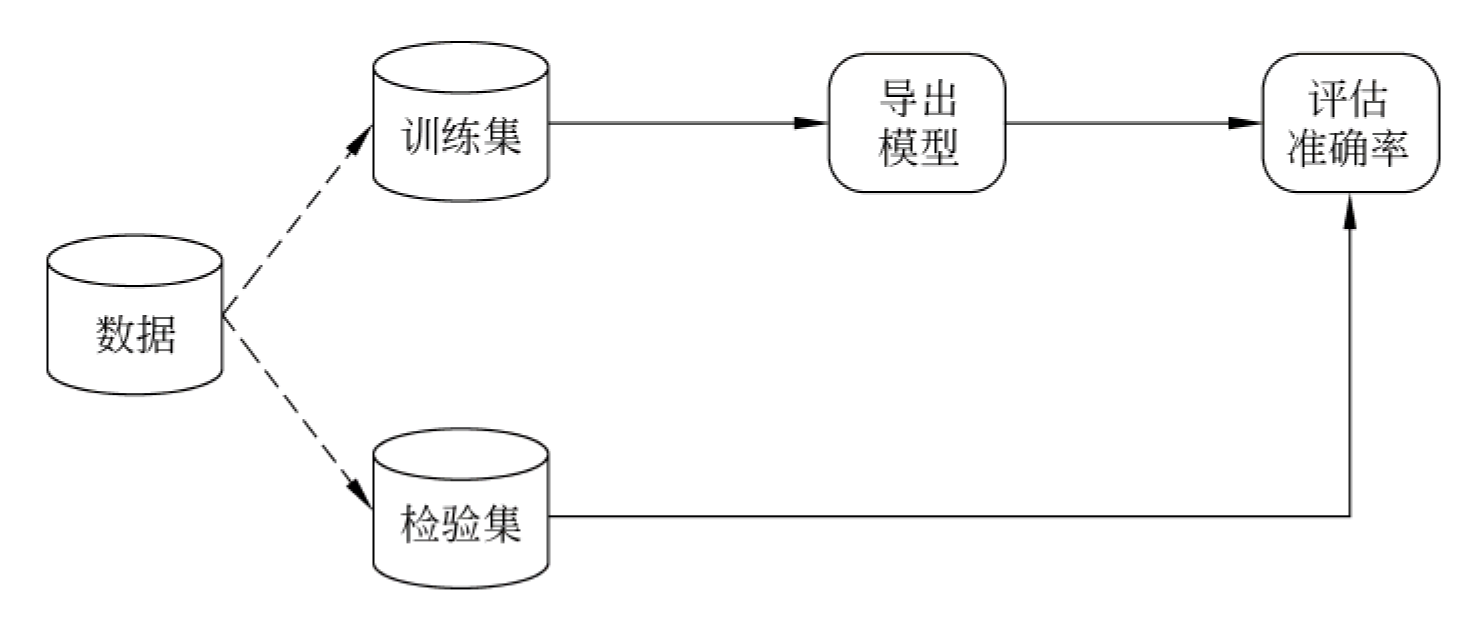


图 5-30 用保持方法估计准确率

随机二次抽样(random subsampling)是保持方法的一种变形，它将保持方法重复 k 次。总准确率估计取每次迭代准确率的平均值。

5.6.3 交叉验证

在 k 折交叉验证(k-fold cross-validation)中，初始数据被随机地划分成 k 个互不相交的子集或"折"$D_1,D_2,\cdots,D_k$，每个折的大小大致相等。训练和检验进行 k 次。在第 i 次迭代，分区 D_i 用做检验集，其余的分区一起用做训练模型。也就是说，在第一次迭代，子集 $D_2,D_3,\cdots,D_k$ 一起作为训练集，得到第一个模型，并在 D_1 上检验；第二次迭代，在子集 $D_1,D_3,D_4,\cdots,D_k$ 上训练，并在 D_2 上检验；如此下去。与上面的保持和随机二次抽样不同，这里的每个样本用于训练的次数相同，并且用于检验一次。对于分类，准确率估计是 k 次迭代正确分类的元组总数除以初始数据中的元组总数。

留一(leave-one-out)是 k 折交叉验证的特殊情况，其中 k 设置为初始元组数。也就是说，每次只给检验集"留出"一个样本。在**分层交叉验证**(stratified corss-validation)中，折被分层，使得每个折中样本的类分布与在初始数据中的类分布大致相同。

一般地，建议使用分层 10 折交叉验证估计准确率(即使计算能力允许使用更多的折)，因为它具有相对较低的偏倚和方差。

5.6.4 自助法

与上面提到的准确率估计方法不同，**自助法**(bootstrap)从给定训练元组中有放回地均匀抽样。也就是说，每当选中一个元组，它等可能地被再次选中并被再次添加到训练集中。例如，想象一台从训练集中随机选择元组的机器。在有放回的抽样中，允许机器多次

选择同一个元组。

有多种自助方法。最常用的一种是“.632”自助法，其方法如下。假设给定的数据集包含 d 个元组。该数据集有放回地抽样 d 次，产生 d 个样本的自助样本集或训练集。原数据元组中的某些元组很可能在该样本集中出现多次。没有进入该训练集的数据元组最终形成检验集。假设进行多次这样的抽样。其结果是，在平均情况下，63.2%的原数据元组将出现在自助样本中，而其余 36.8%的元组将形成检验集。

数字 63.2%从何而来？每个元组被选中的概率是 $1/d$，因此未被选中的概率是 $1-1/d$。需要挑选 d 次，因此一个元组在 d 次挑选都未被选中的概率是 $(1-1/d)^d$。如果 d 很大，该概率近似为 $e^{-1}=0.368$，因此 36.8%的元组未被选为训练元组而留在检验集中，其余 63.2%的元组将形成训练集。

可以重复抽样过程 k 次，在每次迭代中，使用当前的检验集得到从当前自助样本得到的模型的准确率估计。模型的总体准确率则用下式估计：

$$\mathrm{Acc}(M)=\sum_{i=1}^{k}(0.632\times \mathrm{Acc}(M_i)_{\mathrm{test_set}}+0.368\times \mathrm{Acc}(M_i)_{\mathrm{train_set}}) \tag{5-31}$$

其中，$\mathrm{Acc}(M_i)_{\mathrm{test_set}}$是自助样本 i 得到的模型用于检验集 i 的准确率。$\mathrm{Acc}(M_i)_{\mathrm{train_set}}$是自助样本 i 得到的模型用于原数据元组集的准确率。对于小数据集，自助法效果很好。

5.6.5 使用统计显著性检验选择模型

假设已经由数据产生了两个分类模型 M_1 和 M_2。已经进行了 10 折交叉验证，得到了每个的平均错误率。如何确定哪个模型最好？直观地，可以选择具有最低错误率的模型。然而，平均错误率只是对未来数据真实总体上的错误估计。10 折交叉验证试验的错误率之间可能存在相当大的方差。尽管由 M_1 和 M_2 得到的平均错误率看上去可能不同，但是差别可能不是统计显著的。如果两者之间的差别可能只是偶然的，怎么办？本节讨论这些问题。

为了确定两个模型的平均错误率是否存在“真正的”差别，需要使用统计显著性检验。此外，希望得到平均错误率的置信界，以便做出这样的陈述：“对于未来样本的 95%，观测到的均值将不会偏离正、负两个标准差”或者“一个模型比另一个模型好，误差幅度为±4%”。

为了进行统计检验，我们需要什么？假设对于每个模型，我们做了 10 次 10 折交叉验证，每次使用数据的不同 10 折划分。每个划分都独立地抽取。可以分别对 M_1 和 M_2 得到的 10 个错误率取平均值，得到每个模型的平均错误率。对于一个给定的模型，在交叉验证中计算的每个错误率都已看作是来自一种概率分布的不同的独立样本，一般地，它们服从具有 $k-1$ 个自由度的 t 分布，其中 $k=10$(该分布看上去很像正态分布或高斯分布，尽管定义这两个分布的函数很不相同。两个分布都是单峰的、对称的和钟形的)。这使得我们可以做假设检验，其中所使用的显著性检验是 ***t* 检验**，或**研究者的 *t* 检验**(student's *t*-test)。假设这两个模型相同，换言之，两者的平均错误率之差为 0。如果能够拒绝该假设(称为原假设(null hypothesis))，则可以断言两个模型之间的差是统计显著的，在此情况下，可以选择具有较低错误率的模型。

在数据挖掘实践中，通常使用单个检验集，即可能对 M_1 和 M_2 使用相同的检验集。在这种情况下，对于 10 折交叉验证的每一轮，比较 M_1 和 M_2 两个模型(若有多个模型，则每两个模型间都需比较)。也就是说，对于 10 折交叉验证的第 i 轮，使用相同的交叉验证划分得到 M_1 的错误率和 M_2 的错误率。设 $\mathrm{err}(M_1)_i$ 和 $\mathrm{err}(M_2)_i$ 是模型 M_1 和 M_2 在第 i 轮的错误率。对 M_1 的错误率取平均值得到 M_1 的平均错误率，记为 $\overline{\mathrm{err}}M_1$。类似地，可以得到 $\overline{\mathrm{err}}M_2$。两个模型差的方差记为 $\mathrm{var}(M_1-M_2)$。t 检验计算 k 个样本具有 $k-1$ 自由度的 t 统计量。在这个例子中，$k=10$，因为这里的 k 个样本是从每个模型的 10 折交叉验证中得到的错误率。两个模型比较的 t 统计量按下式计算：

$$t=\frac{\overline{\mathrm{err}}(M_1)-\overline{\mathrm{err}}(M_2)}{\sqrt{\mathrm{var}(M_1-M_2)/k}} \tag{5-32}$$

其中：

$$\mathrm{var}(M_1-M_2)=\frac{1}{k}\sum_{i=1}^{k}[\mathrm{err}(M_1)_i-\mathrm{err}(M_2)_i-(\overline{\mathrm{err}}(M_1)_i-\overline{\mathrm{err}}(M_2)_i)]^2 \tag{5-33}$$

为了确定 M_1 和 M_2 是否显著不同，计算 t 并选择**显著水平** sig。在实践中，通常使用 5% 或 1% 的显著水平。然后，在标准的统计学教科书中查找 t 分布表。通常，该表以自由度为行，显著水平为列。假定要确定 M_1 和 M_2 之间的差对总计的 95%(即 sig=5% 或 0.05)是否显著不同。需要从该表查找对应于 $k-1$ 个自由度(对于本例，自由度为 9)的 t 分布值。然而，由于 t 分布是对称的，通常只显示分布上部的百分点。因此，找 $z=\mathrm{sig}/2=0.025$ 的表值，其中 z 也称为**置信界**(confident limit)。如果 $t>z$ 或 $t<-z$，则 t 值落在拒斥域，在分布的尾部。这意味可以拒绝 M_1 和 M_2 的均值相同的原假设，并断言两个模型之间存在统计显著的差别。否则，如果不能拒绝原假设，于是断言 M_1 和 M_2 之间的差可能是随机的。

如果有两个检验集而不是单个检验集，则使用 t 检验的非逐对版本，其中两个模型的均值之间的方差估计为

$$\mathrm{var}(M_1-M_2)=\sqrt{\frac{\mathrm{var}(M_1)}{k_1}+\frac{\mathrm{var}(M_2)}{k_2}} \tag{5-34}$$

其中，k_1 和 k_2 分别用于 M_1 和 M_2 的交叉验证样本数(在本例的情况下，为 10 折交叉验证的轮数)。这也是两个样本的 t 检验。在查 t 分布表时，自由度取两个模型的最小自由度。

5.7 组合分类器技术

组合分类器(ensemble)是一个复合模型，由多个分类器组合而成。个体分类器投票，组合分类器基于投票返回类标号预测。组合分类器比它的成员分类器更准确。

在 5.7.1 节，介绍一般性组合分类方法。后面几节分别介绍了装袋(5.7.2 节)、提升(5.7.3 节)和随机森林(5.7.4 节)。5.7.5 节介绍了提高类不平衡数据分类准确率的方法。

5.7.1 组合分类方法简介

装袋、提升和随机森林都是组合分类方法的例子。组合分类把 k 个学习得到的模型(或基分类器)$M_1, M_2, \cdots, M_k$ 组合在一起,旨在创建一个改进的复合分类模型 M^*。使用给定的数据集 D 创建 k 个训练集 $D_1, D_2, \cdots, D_k$,其中 D_i 用于创建分类器 M_i。给定一个待分类的新数据元组,每个基分类器通过返回类预测投票。组合分类器基于基分类器的投票返回类预测。

组合分类器往往比它的基分类器更准确。理想情况下,基分类器之间几乎不相关。基分类器还应该优于随机猜测。每个基分类器都可以分配到不同的 CPU 上,因此组合分类方法是可并行的。

5.7.2 装袋

先直观地考察装袋(bagging)是如何提高准确率的。假设你是一个病人,希望根据你的症状做出诊断。你可能选择看多个医生,而不是一个。如果某种诊断比其他诊断出现的次数多,则你可能将它作为最终或最好的诊断。也就是说,最终诊断是根据多数表决做出的,其中每个医生都具有相同的投票权重。现在,将医生换成分类器,就可以得到装袋的基本思想。直观地,更多医生的多数表决比少数医生的多数表决更可靠。

给定 d 个元组的集合 D,**装袋**(bagging)过程如下。对于迭代 $i(i=1,2,\cdots,k)$,d 个元组的训练集 D_i 采用有放回抽样,由原始元组集 D 抽取。注意,术语装袋表示自助聚集(bootstrap aggregation)。每个训练集都是一个自助样本。由于使用有放回抽样,D 的某些元组可能不在 D_i 中出现,而其他元组可能出现多次。由每个训练集 D_i 学习,得到一个分类模型 M_i。为了对一个未知元组 X 分类,每个分类器 M_i 返回它的类预测,算作一票。装袋分类器 M^* 统计得票,并将得票最高的类赋予 X。通过取给定检验元组的每个预测平均值,装袋也可以用于连续值的预测。算法如图 5-31 所示。

```
算法5-2：装袋算法——为学习方案创建组合分类模型，其中每个模型给出等权重预测。
输入：
    • D：d个训练元组的集合；
    • k：组合分类器中的模型数；
    • 一种学习方案（例如，决策树算法、后向传播等）
输出：组合分类器——复合模型M*。
方法：
(1) for i=1 to k do//创建k个模型
(2) 通过对D进行有放回抽样，创建自助样本Di；
(3) 使用Di和学习方法导出模型Mi；
(4) endfor

使用组合分类器对元组X分类：
让k个模型都对X分类并返回多数表决；
```

图 5-31 装袋算法

装袋分类器的准确率通常显著高于从原训练集 D 导出的单个分类器的准确率。对于噪声数据和过分拟合的影响,它也不会很差并且更鲁棒。准确率的提高是因为复合模

型降低了个体分类器的方差。

5.7.3 提升和 AdaBoost

现在考察组合分类方法提升。与 5.7.2 节一样，假设你是一位患者，有某些症状。你选择咨询多位医生，而不是一位。假设你根据医生先前的诊断准确率对每位医生的诊断赋予一个权重，然后，将这些加权诊断的组合作为最终的诊断。这就是提升的基本思想。

在**提升**(boosting)方法中，权重赋予每个训练元组。迭代地学习 k 个分类器。学习得到分类器 M_i 之后，更新权重，使得其后的分类器 M_{i+1}“更关注”M_i 误分类的训练元组。最终提升的分类器 M^* 组合每个个体分类器的表决，其中每个分类器投票的权重是其准确率的函数。

AdaBoost(Adaptive Boosting)是一种流行的提升算法。假设我们想提升某种学习方法的准确率。给定数据集 D，它包含 d 个类标记的元组 $(X_1, y_1), (X_2, y_2), \cdots, (X_d, y_d)$，其中 y_i 是元组 X_i 的类标号。开始，AdaBoost 对每个训练元组赋予相等的权重 $1/d$。为组合分类器产生 k 个基分类器需要执行算法的其余部分 k 轮。在第 i 轮，从 D 中元组抽样，形成大小为 d 的训练集 D_i。使用有放回抽样——同一个元组可能被选中多次。每个元组被选中的机会由它的权重决定。从训练集 D_i 导出分类器 M_i，然后使用 D_i 作为检验集计算 M_i 的误差。训练元组的权重根据它们的分类情况调整。

如果元组分类不正确，则它的权重增加。如果元组分类正确，则它的权重减少。元组的权重反映对它们分类的困难程度——权重越高，越可能错误地分类。然后，使用这些权重，为下一轮的分类器产生训练样本。其基本思想是，当建立分类器时，希望它更关注上一轮错误分类的元组。某些分类器对某些“困难”元组分类可能比其他分类器好。这样，建立了一个互补的分类器系列。

现在考察该算法涉及的某些数学问题。为了计算模型 M_i 的错误率，求 M_i 误分类 D_i 中的每个元组的加权和：

$$\mathrm{error}(M_i) = \sum_{j=1}^{d} w_i \times \mathrm{err}(X_j) \tag{5-35}$$

其中 $\mathrm{err}(X_j)$是元组 X_j的误分类误差：如果 X_j被误分类，则 $\mathrm{err}(X_j)$为 1；否则，它为 0。如果分类器 M_i 的性能太差，错误率超过 0.5，则丢弃它，并重新产生新的训练集 D_i，由它导出新的 M_i。

M_i 的错误率影响训练元组权重的更新。如果一个元组在第 i 轮正确分类，则其权重乘以 $\mathrm{error}(M_i)/(1-\mathrm{error}(M_j))$。一旦所有正确分类元组的权重都被更新，就对所有元组的权重(包括误分类的元组)规范化，使得它们的和与以前一样。为了规范化权重，将它乘以旧权重之和。结果，正如上面介绍的一样，误分类元组的权重增加，而正确分类元组的权重减少。

一旦提升完成，如何使用分类器的组合预测元组 X 的类标号？与装袋将相同的表决权赋予每个分类器的方法不同，提升根据分类器的分类情况，对每个分类的表决权赋予一个权重。分类器的错误率越低，它的准确率就越高，因此它的表决权重就应当越高。分类器 M_i 的表决权重为

$$\log \frac{1-\mathrm{error}(M_i)}{\mathrm{error}(M_i)} \tag{5-36}$$

对于每个类 c，求每个将类 c 指派给 X 的分类器的权重之和。具有最大权重和的类是“赢家”，并返回作为元组 X 的类预测。

由于提升关注误分类元组，所以存在结果复合模型对数据过分拟合的危险。因此，提升的模型有时可能没有从相同数据导出的单一模型的准确率高。装袋不太受过分拟合的影响。尽管与单个模型相比，提升与装袋都能够显著提高准确率，但是提升往往得到更高的准确率。

5.7.4 随机森林

本节介绍一种组合方法，称为**随机森林**。想象组合分类器中的每个分类器都是一棵决策树，因此分类器的集合就是一个森林。个体决策树在每个节点使用随机选择的属性决定划分。更准确地说，每一棵树都依赖于独立抽样，并与森林中所有树具有相同分布的随机向量的值。分类时，每棵树都投票并且返回得票最多的类。

随机森林可以使用装袋与随机属性选择结合来构建。给定 d 个元组的训练集 D，组合分类器产生 k 棵决策树的一般过程如下。对于每次迭代 $i(i=1,2,\cdots,k)$，使用有放回抽样，由 D 产生 d 个元组的训练集 D_i。也就是说，每个 D_i 都是 D 的一个自助样本，使得某些元组可能在 D_i 出现多次，而另一些可能不出现。设 F 是用来在每个节点决定划分的属性数，其中 F 远小于可用属性数。为了构造决策树分类器 M_i，在每个节点随机选择 F 个属性作为该节点划分的候选属性。使用 CART 算法的方法使树增长到最大规模，并且不剪枝。用这种方式，使用随机输入选择形成的随机森林称为 Forest-RI。

随机森林的另一种形式称为 Forest-RC，使用输入属性的随机线性组合。它不是随机地选择一个属性子集，而是由已有属性的线性组合创建一些新属性（特征），即一个属性由指定的 L 个原属性组合产生。在每个给定的节点，随机选择 L 个属性，并且以从$[-1,1]$中随机选取的数为系数相加，产生 F 个线性组合，并在其中搜索找到最佳划分。当只有少量属性可用时，为了降低个体分类器之间的相关性，可以采用这种形式的随机森林。

随机森林的准确率可以与 AdaBoot 相媲美，但是对错误和离群点更鲁棒。随着森林中树的个数增加，森林的泛化误差收敛。因此，过拟合不是问题。随机森林的准确率依赖于个体分类器的实例和它们之间的依赖性。理想情况是保持个体分类器的能力而不是提高它们的相关性。随机森林对每次划分所考虑的属性数很敏感。通常选取 $\log_2 d+1$ 个属性。由于随机森林在每次划分时只考虑很少的属性，因此它们在大型数据库上非常有效。它们可能比装袋和提升更快。随机森林给出了变量重要性的内在估计。

5.7.5 提高类不平衡数据的分类准确率

本节再次考虑类不平衡问题，主要研究提高类不平衡数据分类准确率的方法。

给定两类数据，该数据是类不平衡的，如果感兴趣的主类（正类）只有少量元组代表，而大多数元组都代表负类。对于多类不平衡数据，每个类的数据分布差别显著，其中，主类或感兴趣的类的元组稀少。类不平衡问题与代价敏感学习密切相关，那里每个类的错

误代价并不相等。例如,在医疗诊断中,错误地把一位癌症患者诊断为健康(假阴性)的代价远高于错误地把一个健康人诊断为患有癌症(假阳性)。假阴性错误可能导致失去生命,因此比假阳性错误的代价高得多。类不平衡数据的其他应用包括欺诈检测、从卫星雷达图像检测石油泄漏和故障检测。

传统的分类算法旨在最小化分类误差。它们假定:假正例和假负例错误的代价是相等的。由于假定类平衡分布和相等的错误代价,所以传统的分类算法不适合类不平衡数据。尽管准确率度量假定各类的代价都相等,但是可以使用不同类型分类的其他评估度量。

本节考察提高类不平衡数据分类准确率的一般方法。这些方法包括过抽样、欠抽样、阈值移动、组合技术。前3种不涉及对分类模型结构的改变。也就是说,过抽样和欠抽样改变训练集中的元组分布,阈值移动影响对新数据分类时模型如何决策。为了便于解释,我们针对两类不平衡数据问题介绍一般方法,其中代价较高的类比代价较低的类稀少。

过抽样和欠抽样都改变训练集的分布,使得稀有(正)类能够很好地表示。过抽样对正元组重复采样,使得训练集包含相同个数的正元组和负元组。欠抽样减少负元组的数量,它随机地从多数(负)类中删除元组,直到正元组和负元组的数量相等。

不平衡类问题的**阈值移动**(threshold-moving)方法不涉及抽样。它用于对给定输入元组返回一个连续输出值的分类器。即对于输入元组 X,这种分类器返回一个映射 $f(X)\to[0,1]$作为输出。该方法不是操控训练元组,而是基于输出值返回分类决策。最简单的方法是,对于某个阈值 t,满足 $f(X)\geqslant t$ 的元组 X 被视为正的,而其他元组被看做负的。其他方法可能涉及用加权操控输出。一般而言,阈值移动方法移动阈值 t,使得稀有类的元组容易分类。阈值移动方法尽管不像过抽样和欠抽样那么流行,但是它简单,并且对于两类不平衡数据已经表现得相当成功。

组合方法也已经用于类不平衡问题。组成组合分类器的个体分类器可以使用上面介绍的方法,如过抽样和阈值移动。

上面介绍的方法对两类任务的类不平衡问题相对有效。试验观察表明,阈值移动和组合方法优于过抽样和欠抽样。即便在非常不平衡的数据集上,阈值移动也很有效。多类任务上的类不平衡困难得多,那里过抽样和阈值移动都不太有效果。尽管阈值移动和组合方法表现出了希望,但是为多类不平衡问题寻找更好的解决方案仍然是尚待解决的问题。

5.8 惰性学习法(k最近邻分类)

当给定训练元组集时,急切学习法(eager learning)在接收待分类的新元组之前就构造泛化模型(即分类模型)。可以认为学习后的模型已经就绪,并急于对先前未见过的元组进行分类。而惰性方法的学习过程直到对给定的检验元组分类之前的一刻才构造模型。也就是说,当给定一个训练元组时,惰性学习法(lazy learning)简单地存储它,并且一直等待,直到给定一个检验元组。本节介绍一个惰性学习算法——k 最近邻分类。

最近邻分类法是基于类比学习,即通过将给定的检验元组与和它相似的训练元组进

行比较来学习。训练元组用 n 个属性描述。每个元组代表 n 维空间的一个点。这样，所有的训练元组都存放在 n 维模式空间中。当给定一个未知元组时，k 最近邻分类（k-nearest-neighbor classification）搜索模式空间，找出最接近未知元组的 k 个训练元组。这 k 个训练元组是未知元组的 k 个最近邻。

邻近性用距离度量，如欧几里得距离。两个点或元组 $X_1=(x_{11},x_{12},\cdots,x_{1n})$ 和 $X_2=(x_{21},x_{22},\cdots,x_{2n})$ 的欧几里得距离是

$$\mathrm{dist}(X_1,X_2)=\sqrt{\sum_{i=1}^{n}(x_{1i}-x_{2i})^2} \tag{5-37}$$

换言之，对于每个数值属性，取元组 X_1 和 X_2 该属性对应值的差，取差的平方和，并取其平方根。通常，在使用式(5-37)之前，要把每个属性的值规范化，这有助于防止具有较大初始值域的属性（如收入）比具有较小初始值域的属性（如二元属性）的权重过大。例如，可以通过计算下式，使用最小-最大规范化把输指数型 A 的值 v 变换到[0,1]区间中的 v'

$$v'=\frac{v-\min_A}{\max_A-\min_A} \tag{5-38}$$

其中，$\min_A$ 和 $\max_A$ 分别是属性 A 的最小值和最大值。

对于 k 最近邻分类，未知元组被指派到它的 k 个最近邻中的多数类。当 $k=1$ 时，未知元组被指派到模式空间中最接近它的训练元组所在的类。最近邻分类也可以用于数值预测，即返回给定未知元组的实数值预测。在这种情况下，分类器返回未知元组的 k 个最近邻的实数值标号的平均值。

上面的讨论假定用来描述元组的属性都是数值的。对于标称属性，一种简单的方法是比较元组 X_1 和 X_2 中对应属性的值。如果二者相同，则二者之间的差为 0；如果二者不同，则二者之间的差为 1。

通常，如果元组 X_1 或 X_2 在给定属性 A 上的值缺失，则假定取最大的可能差。假设每个属性都已经映射到[0,1]区间。对于标称属性，如果 A 的一个或两个对应值缺失，则取差值为 1。如果 A 是数值属性，并且在元组 X_1 和 X_2 上都缺失，则差值也取 1。如果只有一个值缺失，而另一个存在并且已经规范化，则取差为 $|1-v'|$ 和 $|0-v'|$ 中的最大者。

近邻数 k 的值可以通过实验来确定，从 $k=1$ 开始，使用检验集估计分类器的错误率。重复该过程，每次 k 增 1，允许增加一个近邻。可以选取产生最小错误率的 k。一般而言，训练元组越多，k 的值越大。随着训练元组数趋向于无穷并且 $k=1$，错误率不会超过贝叶斯错误率的 2 倍。如果 k 也趋向于无穷，则错误率趋向于贝叶斯错误率。

最近邻分类法使用基于距离的比较，本质上赋予每个属性相等的权重。因此，当数据存在噪声或不相关属性时，它们的准确率可能受到影响。然而，这种方法已经被改进，结合属性加权和噪声数据元组的剪枝。距离度量的选择可能是至关重要的。也可以使用曼哈顿距离或其他距离度量。

最近邻分类法在对检验元组分类时可能非常慢。如果 D 是由 $|D|$ 个元组组成的训练数据库，而 $k=1$，则对一个给定的检验元组分类需要 $O(|D|)$ 次比较。通过预先排序并将排序后的元组安排在搜索树中，比较次数可以降低到 $O(\log|D|)$。并行实现可以把运行

时间降低为常数，即 $O(1)$，独立于 $|D|$。

加快分类速度的其他技术包括使用部分距离计算和编辑存储的元组。**部分距离**(partial distance)方法基于 n 个属性的子集计算距离。如果该距离超过阈值，则停止给定存储元组的进一步计算，该过程转向下一个存储元组。**编辑**(editing)方法可以删除被证明"无用的"元组。该方法也称**剪枝**或**精简**，因为它减少了存储元组的总数。

5.9 基于 Python 平台的案例分析

本节通过一个例子讲解决策树如何预测患者需要佩戴的隐形眼镜类型。使用小数据集，就可以利用决策树学到很多知识：眼科医生是如何判断患者需要佩戴的镜片类型的；一旦理解了决策树的工作原理，我们甚至也可以帮助人们判断需要佩戴的镜片类型。

5.9.1 数据集准备

隐形眼镜数据集是非常著名的数据集，它包含很多患者眼部状况的观察条件以及医生推荐的隐形眼镜类型。隐形眼镜类型包括硬材质、软材质以及不适合佩戴隐形眼镜。部分数据集如表 5-5 所示。

表 5-5 隐形眼镜数据集

age	prescript	astigmatic	tearRate	type
young	myope	no	reduced	no_lenses
young	myope	no	normal	soft
young	myope	yes	reduced	no_lenses
young	myope	yes	normal	hard
young	hyper	no	reduced	no_lenses
young	hyper	no	normal	soft
young	hyper	yes	reduced	no_lenses
young	hyper	yes	normal	hard
pre	myope	no	reduced	no_lenses
pre	myope	no	normal	soft
pre	myope	yes	reduced	no_lenses
pre	myope	yes	normal	hard
pre	hyper	no	reduced	no_lenses
pre	hyper	no	normal	soft
pre	hyper	yes	reduced	no_lenses
pre	hyper	yes	normal	no_lenses

续表

age	prescript	astigmatic	tearRate	type
presbyopic	myope	no	reduced	no_lenses
presbyopic	myope	no	normal	no_lenses
presbyopic	myope	yes	reduced	no_lenses
presbyopic	myope	yes	normal	hard
presbyopic	hyper	no	reduced	no_lenses
presbyopic	hyper	no	normal	soft
presbyopic	hyper	yes	reduced	no_lenses
presbyopic	hyper	yes	normal	no_lenses

5.9.2 算法描述

本节将通过 ID3 算法一步步地构造决策树，首先讨论数学上如何使用信息论划分数据集，然后编写代码将理论应用到具体的数据集上，最后编写代码构建决策树。

在构造决策树时，需要解决的第一个问题就是，当前数据集上哪个特征在划分数据分类时起决定性作用。为了找到决定性的特征，划分出最好的结果，必须评估每个特征。完成测试之后，原始数据集就被划分为几个数据子集。这些数据子集会分布在第一个决策点的所有分枝。如果某个分枝下的数据属于同一类型，则生成一个新的叶子节点。如果数据子集内的数据不属于同一类型，则需要重复划分数据子集的过程。划分数据子集的算法和划分原始数据集的方法相同，直到所有具有相同类型的数据均在一个数据子集内。

划分数据集的最大原则是：将无序的数据集变得更加有序。可以使用多种方法划分数据集，但是每种方法都有各自的优缺点。组织杂乱无章数据的一种方法就是使用信息论度量信息，信息论是量化处理信息的分支科学。可以在划分数据之前或之后使用信息论量化度量信息的内容，在划分数据集之前和之后信息发生的变化称为信息增益，知道如何计算信息增益，就可以计算每个特征值划分数据集获得的信息增益，获得信息增益最高的特征就是最好的选择。为了精确地定义信息增益，先定义信息论中广泛使用的一个度量标准——香农熵，或者简称为熵，它刻画了任意样例集的纯度。

熵定义为信息的期望值，在明晰这个概念之前，必须知道信息的定义。如果待分类的事务可能划分在多个分类之中，则符号 x_i 的信息定义为

$$l(x_i) = -\log_2 p(x_i) \tag{5-39}$$

其中 $p(x_i)$是选择该分类的概率。

为了计算熵，需要计算所有类别所有可能值包含的信息期望值，通过下面的公式得到：

$$H = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) \tag{5-40}$$

其中 n 是分类的数目。

下面学习如何使用 Python 计算信息熵,如图 5-32 所示。

```
#计算给定数据集的熵
def calc ShannonEnt(dataSet):
      numEntries=len(dataSet)
      labelCounts={}
      for featVec in dataSet:
          currentLabel=featVec[-1]
          if currentLabel not in labelCounts.keys():
             labelCounts[currentLabel]=0
          labelCounts[currentLabel]+=1
      shannonEnt=0.0
      for key in labelCounts:
          prob=float(labelCounts[key])/numEntries
          shannonEnt-=prob*log(prob, 2)
      return shannonEnt
```

图 5-32 计算信息熵命令

得到熵以后,就可以按照获取最大信息增益的方法找到最佳特征去划分数据集,遍历整个数据集,用每个特征对数据集进行划分,然后计算每种划分方式的信息熵,与划分之前的信息熵进行对比,找到信息增益最大的划分方式,确定最佳划分特征。程序代码如图 5-33 所示。

```
#选择最好的数据集划分方式
def choooseBestFeatureToSplit(dataSet):
      numFeatures=len(dataSet[0])-1
      baseEntropy=calcShannonEnt(dataSet)
      bestInfoGain=0.0; bestFeature=-1
      for i in range(numFeatures):#控制特征, 找到最好的特征划分
          featList=[example[i] for example in dataSet]
          uniqueVals=set(featList) #set()集合函数
          newEntropy=0.0
          for value in uniqueVals:
             subDataSet=splitDataSet(dataSet, i, value)
             prob=len(subDataSet)/float(len(dataSet))
             newEntropy+=prob* calcShannonEnt(subDataSet)
          infoGain=baseEntropy-newEntropy
      if(infoGain>bestInfoGain):
          bestInfoGain=infoGain
          bestFeature=i
      return bestFeature
```

图 5-33 选择数据集的划分方式

前面已经学习了从数据集构造决策树算法所需要的子功能模块,其工作原理如下:得到原始数据集,然后基于最好的属性值划分数据集,由于特征值可能多于两个,因此可能存在多于两个分枝的数据集划分。第一次划分完成以后,数据将被向下传递到树分枝的下一个节点,在这个节点上可以再次划分数据,因此可以采用递归的原则处理数据集。

递归结束的条件是:程序遍历完所有划分数据集的属性,或者每个分枝下的所有实例都具有相同分类。如果所有实例具有相同的分类,则得到一个叶子节点或者终止块。任何到达叶子点的数据必然属于叶子节点的分类。如果数据集已经处理了所有属性,但是类标签依然不是唯一的,此时需要决定该叶子节点的分类,在这种情况下,通常会采用

多数表决的方法决定该叶子节点的分类。使用 Python 创建决策树的程序代码如图 5-34 所示。

```
#创建决策树
def create Tree(dataSet,labels):
    classList=[example[-1] for example in dataSet]
    if classList.count(classList[0])==len(classList):#count()计算出现次数
            return classList[0] 4: #遍历完所有特征时返回出现次数最多的
          if len(dataSet[0])==1:
            return majorityCnt(classList)
    bestFeat=chooseBestFeatureToSplit(dataSet)
    bestFeatLabel=labels[bestFeat]
    myTree={bestFeatLabel:{}}#建立以bestFeatLabel为跟的空子树
    del(labels[bestFeat])
    featValues=[example[bestFeat] for example in dataSet]
    uniqueVals=set(featValues)
    for value in uniqueVals:
            subLabels=labels[:]
            myTree[bestFeatLabel][value]=createTree\
                  (splitDataSet(dataSet, bestFeat, value), subLabels)
    return myTree
```

图 5-34　创建决策树

5.9.3　算法测试

为了方便读者阅读，将这个程序集成到一个决策树系统中，如图 5-35 所示。选择需要训练的数据集，然后输入需要的属性名称，最终生成决策树，并给出决策树面对测试集的分类准确率。



图 5-35　决策树算法的界面

图 5-36 给出了一个用字典表示的决策树，采用文本方式很难分辨出决策树的模样，决策树的主要优点就是直观易于理解，如果不能将其直观地显示出来，就无法发挥其优势。

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Administrator\Desktop\python决策树算法\App_D_tree.py =====
['age', 'prescript', 'astigmatic', 'tearRate']
{'tearRate': {'normal': {'astigmatic': {'yes': {'prescript': {'hyper': {'age': {'presbyopic': 'no_lenses', 'young': '
hard', 'pre': 'no_lenses'}}, 'myope': 'hard'}}, 'no': {'age': {'presbyopic': {'prescript': {'hyper': 'soft', 'myope': '
no_lenses'}}, 'young': 'soft', 'pre': 'soft'}}}}, 'reduced': 'no_lenses'}}
```

图 5-36　决策树程序运行的结果

为了直观地显示决策树，用 Matplotlib 库编写绘制树的程序，在图 5-37 中，我们可以看到一个非常直观的决策树，沿着决策树的不同分枝，可以得到不同患者需要佩戴的隐形眼镜类型。

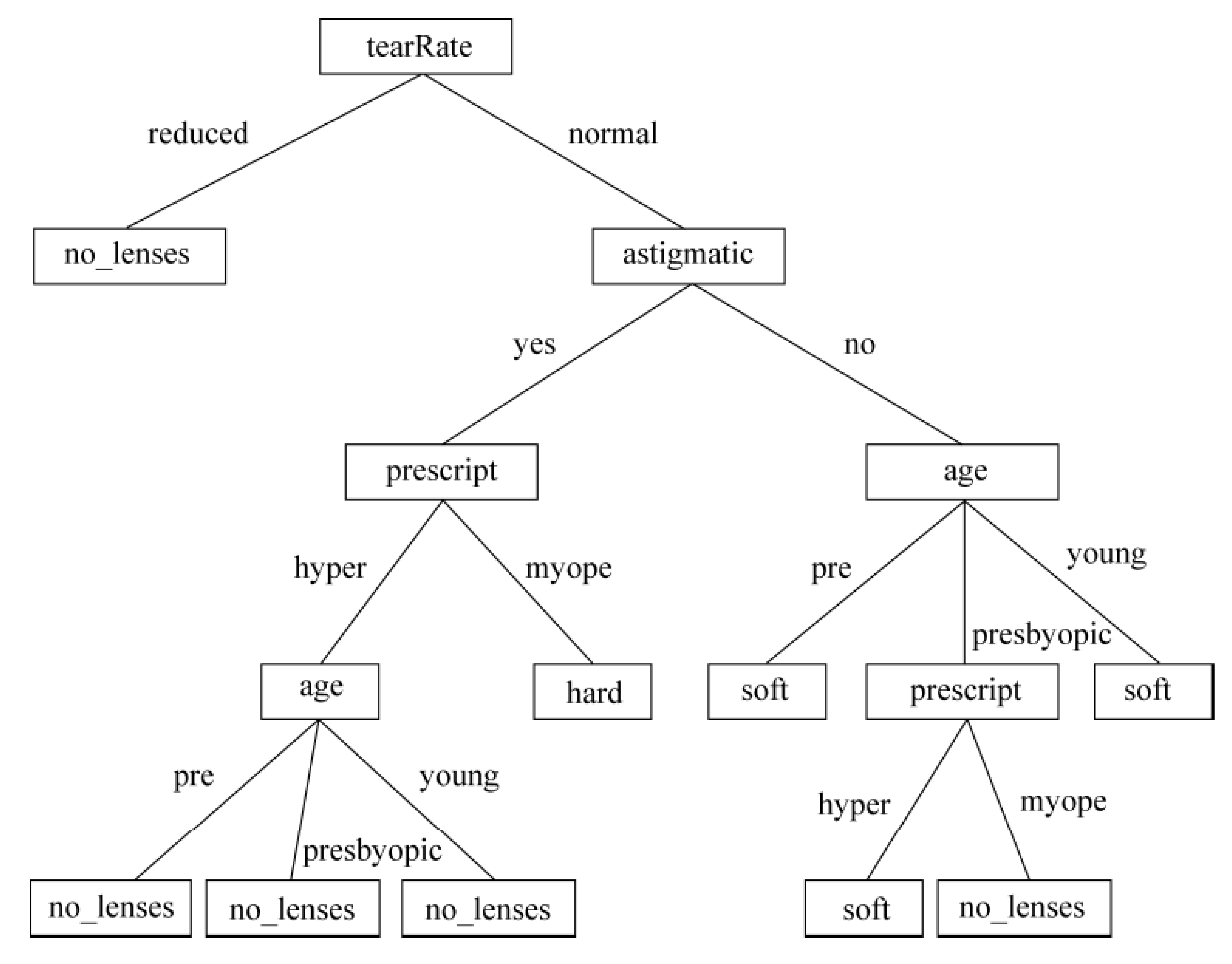


图 5-37 由 ID3 算法产生的决策树

为了验证决策树对未知数据的预测性能，把隐形眼镜数据集随机划分成两部分：一部分样本构成训练集，如表 5-6 所示；另一部分样本构成测试集，如表 5-7 所示。

表 5-6 新构成的训练集

age	prescript	astigmatic	tearRate	type
young	myope	no	reduced	no_lenses
young	myope	no	normal	soft
young	myope	yes	reduced	no_lenses
young	hyper	no	reduced	no_lenses
young	hyper	no	normal	soft
young	hyper	yes	reduced	no_lenses
pre	myope	no	reduced	no_lenses
pre	myope	no	normal	soft
pre	myope	yes	reduced	no_lenses
pre	hyper	no	reduced	no_lenses
pre	hyper	no	normal	soft

续表

age	prescript	astigmatic	tearRate	type
pre	hyper	yes	reduced	no_lenses
presbyopic	myope	no	reduced	no_lenses
presbyopic	myope	no	normal	no_lenses
presbyopic	myope	yes	reduced	no_lenses
presbyopic	hyper	no	reduced	no_lenses
presbyopic	hyper	no	normal	soft
presbyopic	hyper	yes	reduced	no_lenses

表 5-7　新构成的测试集

age	prescript	astigmatic	tearRate	type
presbyopic	hyper	yes	normal	no_lenses
young	hyper	yes	reduced	no_lenses
presbyopic	myope	no	normal	no_lenses
presbyopic	myope	yes	reduced	no_lenses
presbyopic	hyper	no	reduced	no_lenses
pre	myope	no	normal	soft
pre	myope	yes	reduced	no_lenses
pre	hyper	no	reduced	no_lenses
pre	hyper	no	normal	soft

依靠训练集数据构造决策树之后，可以将它用于实际数据的分类。在执行数据分类时，需要决策树以及用于构造树的属性。然后，系统比较待测数据与决策树上的数值，递归执行该过程，直到进入叶子节点。

如图 5-38 所示，用测试集去测试训练集训练的决策树，准确度可以达到 89%。读者也可以自己划分训练集和测试集来验证决策树的准确度。另外，这个决策树算法没有进行剪枝，读者可以尝试优化这个算法。



图 5-38　计算决策树的准确率

5.10 小结

- **分类**是一种数据分析形式，它提取描述数据类的模型。分类器或分类预测类别标号(类)。**数值预测**建立连续值函数模型。分类和数值预测是两类主要的预测问题。
- **决策树归纳**是一种自顶向下递归树归纳算法，它使用一种属性选择度量为树的每个非树叶节点选择测试属性。**ID3**、**C4.5** 和 **CART** 都是这种算法的例子，它们使用不同的属性选择度量。**树剪枝**算法试图通过剪去反映数据中的噪声的分枝来提高准确率。早期的决策树算法通常假定数据是驻留内存的。
- **朴素贝叶斯分类**基于后验概率的贝叶斯定理。它假定类条件独立——一个属性值对给定类的影响独立于其他属性的值。
- **支持向量机**(SVM)是一种用于线性和非线性数据的分类算法。它把源数据变换到较高维空间，使用称作**支持向量**的基本元组，从中发现分离数据的超平面。
- **混淆矩阵**可以用来评估分类器的质量。对于二元分类问题，它显示真正例、真负例、假正例、假负例。评估分类器预测能力的度量包括**准确率**、**灵敏度**(又称为召回率)、**特效性**、**精度**、$\boldsymbol{F}$ 和 $\boldsymbol{F_\beta}$。当感兴趣的主类占少数时，过分依赖准确率度量可能受骗。
- 分类器的构造与评估需要把标记的数据集划分成训练集和检验集。**保持**、**随机抽样**、**交叉验证**和**自助法**都是用于这种划分的典型方法。
- **组合方法**可以通过学习和组合一系列个体(基)分类器模型来提高总体准确率。**装袋**、**提升**和**随机森林**都是流行的组合方法。
- 当感兴趣的主类只由少量元组代表时就会出现类**不平衡问题**。处理这一问题的策略包括**过抽样**、**欠抽样**、**阈值移动**和**组合技术**。

5.11 习题

1. 简述决策树分类的主要步骤。

2. 给定决策树，选项有：①将决策树转换成规则，然后对结果规则剪枝；②对决策树剪枝，然后将剪枝后的树转换成规则。与②相比，①的优点是什么？

3. 计算决策树算法在最坏情况下的时间复杂度是重要的。给定数据集 D，具有 m 个属性和 $|D|$ 个训练记录，证明决策树生长的计算时间最多为 $mD\log D$。

4. 证明：将节点划分为更小的后续节点之后，节点熵不会增加。

5. 为什么朴素贝叶斯称为“朴素”？简述朴素贝叶斯分类的主要思想。

6. 为 4 个布尔属性 A、B、C 和 D 的奇偶函数画一棵完全决策树。可以简化该决策树吗？

7. 考虑表 5-8 中二元分类问题的训练样本。

(1) 计算整个训练样本集的 Gini 指标值。

(2) 计算属性“顾客 ID”的 Gini 指标值。

(3) 计算属性“性别”的 Gini 指标值。

(4) 计算使用多路划分属性“车型”的 Gini 指标值。

(5) 计算使用多路划分属性“衬衣尺码”的 Gini 指标值。

(6) 下面哪个属性更好：性别、车型还是衬衣尺码？

(7) 解释为什么属性“顾客 ID”的 Gini 值最低，却不能作为属性测试条件。

表 5-8　习题 7 中的训练样本

顾客 ID	性别	车型	衬衣尺码	类
1	男	家用	小	C0
2	男	运动	中	C0
3	男	运动	中	C0
4	男	运动	大	C0
5	男	运动	加大	C0
6	男	运动	加大	C0
7	女	运动	小	C0
8	女	运动	小	C0
9	女	运动	中	C0
10	女	豪华	大	C0
11	男	家用	大	C1
12	男	家用	加大	C1
13	男	家用	中	C1
14	男	豪华	加大	C1
15	女	豪华	小	C1
16	女	豪华	小	C1
17	女	豪华	中	C1
18	女	豪华	中	C1
19	女	豪华	中	C1
20	女	豪华	大	C1

8. 设计一种方法，对无限的数据流进行有效的朴素贝叶斯分类(即只能扫描数据流一次)。如果想发现这种分类模式的演变(例如，将当前的分类模式与较早的模式进行比较，如与一周以前的模式相比)，你有何修改建议？

9. 证明准确率是灵敏性和特效性度量的函数。

10. 概述处理类不平衡问题的方法。假设银行要开发一个分类器以预防信用卡交易中的欺诈。说明基于大量非欺诈实例和很少的欺诈实例如何构造高质量的分类。

5.12 参考文献

[1] Ranka S, Singh V. CLOUDS: A Decision Tree Classifier for Large Datasets[C]//Proceedings of the 4th Knowledge Discovery and Data Mining Conference. 1998: 2-8.

[2] Bishop C M. Neural Networks for Pattern Recognition[M]. Oxford University Press, 1995.

[3] Breiman L, Friedman J, Stone C J, et al. Classification and Regression Trees[M]. CRC press, 1984.

[4] Breslow L A, Aha D W. Simplifying Decision Trees: A Survey[J]. The Knowledge Engineering Review, 1997, 12(01): 1-40.

[5] Buntine W. Learning Classification Trees[J]. Statistics and Computing, 1992, 2(2): 63-73.

[6] Cantú-Paz E, Kamath C. Using Evolutionary Algorithms to Induce Oblique Decision Trees[C]// Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation. Morgan Kaufmann Publishers Inc., 2000: 1053-1060.

[7] Efron B, Tibshirani R J. Cross-Validation and the Bootstrap: Estimating the Error rate of a Prediction Rule[M]. Division of Biostatistics, Stanford University, 1995.

[8] Esposito F, Malerba D, Semeraro G, et al. A Comparative Analysis of Methods for Pruning Decision Trees[J]. IEEE transactions on Pattern Analysis and Machine Intelligence, 1997, 19(5): 476-491.

[9] Fisher R A. The Use of Multiple Measurements in Taxonomic Problems[J]. Annals of Eugenics, 1936, 7(2): 179-188.

[10] Jain A K, Duin R P W, Mao J. Statistical Pattern Recognition: A Review[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(1): 4-37.

[11] Kulkarni S R, Lugosi G, Venkatesh S S. Learning Pattern Classification—A Survey[J]. IEEE Transactions on Information Theory, 1998, 44(6): 2178-2206.

[12] Duda R O, Hart P E, Stork D G. Pattern Classification[M]. John Wiley & Sons, 2012.

[13] Fukunaga K. Introduction to Statistical Pattern Recognition[M]. Academic press, 2013.

[14] Cherkassky V, Mulier F M. Learning from Data: Concepts, Theory, and Methods[M]. John Wiley & Sons, 2007.

[15] Trevor H, Robert T, Jerome F. The Elements of Statistical Learning: Data Mining, Inference and Prediction[J]. New York: Springer-Verlag, 2001, 1(8): 371-406.

[16] Michie D, Spiegelhalter D J, Taylor C C. Machine Learning, Neural and Statistical Classification [J]. 1994.

[17] Mitchell T M. Machine Learning. WCB[J]. 1997.

[18] Moret B M E. Decision Trees and Diagrams[J]. ACM Computing Surveys (CSUR), 1982, 14(4): 593-623.

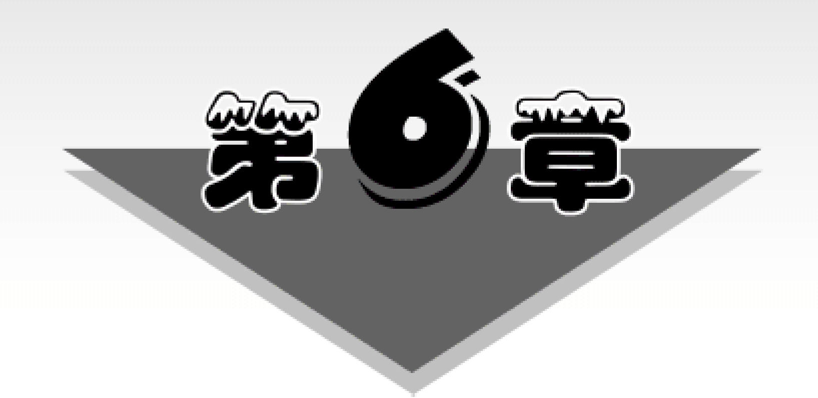
[19] Murthy S K. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey [J]. Data Mining and Knowledge Discovery, 1998, 2(4): 345-389.

[20] Safavian S R, Landgrebe D. A Survey of Decision Tree Classifier Methodology[J]. 1990.

[21] Quinlan J R. Discovering Rules by Induction from Large Collections of Examples[M]. Expert Systems in the Micro Electronic age. Edinburgh University Press, 1979.

[22] Quinlan J R. C4. 5: Programs for Machine Learning[M]. Elsevier, 2014.
[23] Kass G V. An Exploratory Technique for Investigating Large Quantities of Categorical Data[J]. Applied Statistics, 1980: 119-127.
[24] Cover T, Hart P. Nearest Neighbor Pattern Classification[J]. IEEE Transactions on Information Theory, 1967, 13(1): 21-27.
[25] Aha D W. A Study of Instance-based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations[J]. 1990.
[26] Cost S, Salzberg S. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features [J]. Machine Learning, 1993, 10(1): 57-78.
[27] Han E H S, Karypis G, Kumar V. Text Categorization Using Weight Adjusted k-nearest Neighbor Classification[C]//Pacific-asia conference on knowledge discovery and data mining. Springer Berlin Heidelberg, 2001: 53-65.
[28] Langley P, Iba W, Thompson K. An Analysis of Bayesian Classifiers[C]//Aaai. 1992, 90: 223-228.
[29] Ramoni M, Sebastiani P. Robust Bayes Classifiers[J]. Artificial Intelligence, 2001, 125(1): 209-226.
[30] Lewis D D. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval[C]// European Conference on Machine Learning. Springer Berlin Heidelberg, 1998: 4-15.
[31] Domingos P, Pazzani M. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss [J]. Machine learning, 1997, 29(2-3): 103-130.
[32] Heckerman D. Bayesian Networks for Data Mining[J]. Data Mining and Knowledge Discovery, 1997, 1(1): 79-119.
[33] Vapnik V. The Nature of Statistical Learning Theory[M]. Springer Science & Business Media, 2013.
[34] Vapnik V N, Vapnik V. Statistical Learning Theory[M]. New York: Wiley, 1998.
[35] Cristianini N, Shawe-Taylor J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods[M]. Cambridge University Press, 2000.
[36] Scholkopf B, Smola A J. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond[M]. MIT press, 2001.
[37] Burges C J C. A Tutorial on Support Vector Machines for Pattern Recognition[J]. Data Mining and Knowledge Discovery, 1998, 2(2): 121-167.
[38] Bennett K P, Campbell C. Support Vector Machines: Hype or Hallelujah? [J]. ACM SIGKDD Explorations Newsletter, 2000, 2(2): 1-13.
[39] Hearst M A, Dumais S T, Osman E, et al. Support Vector Machines[J]. IEEE Intelligent Systems and their Applications, 1998, 13(4): 18-28.
[40] Mangasarian O L. Data Mining via Support Vector Machines[M]//System Modeling and Optimization XX. Springer US, 2003: 91-112.
[41] Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection [C]//IJCAI. 1995, 14(2): 1137-1145.
[42] Dietterich T G. Ensemble Methods in Machine Learning[C]//International Workshop on Multiple Classifier Systems. Springer Berlin Heidelberg, 2000: 1-15.
[43] Breiman L. Bagging Predictors[J]. Machine Learning, 1996, 24(2): 123-140.

[44] Freund Y，Schapire R E. A Desicion-Theoretic Generalization of On-line Learning and an Application to Boosting[C]//European Conference on Computational Learning Theory. Springer Berlin Heidelberg，1995：23-37.
[45] Breiman L. Bias，Variance，and Arcing Classifiers[J]. 1996.
[46] 刘继宇. 基于粗糙集分类算法的研究及应用[D]. 桂林：广西师范大学，2015.
[47] Jiawei H. 数据挖掘概念与技术[M]. 北京：机械工业出版社，2006.
[48] Pangning T，Steinbach M，Kumar V. 数据挖掘导论. 北京：人民邮电出版社，2006.
[49] 韩家炜，坎伯. 数据挖掘：概念与技术[M]. 北京：机械工业出版社，2001.



数据聚类分析

在对世界的分析和描述中，类，或在概念上有意义的具有公共特性的对象组，扮演着重要的角色。的确，人类擅长将对象划分为组（聚类），并将特定的对象指派到这些组（分类）。例如，即使很小的孩子也能很快地将图片上的对象标记为建筑物、车辆、人、动物、植物等。就理解数据而言，簇是潜在的类，而聚类分析是研究自动发现这些类的技术。

因此，**聚类**是一个把数据对象集划分成多个组或簇的过程，使得簇内的对象具有很高的相似性，但与其他簇中的对象具有很高的相异性。相似性和相异性根据描述对象的属性值来评估，并且通常涉及聚类度量。

聚类分析在许多实际问题上都有应用。下面给出一些具体的例子，以方便读者进行理解。

- **信息检索**。万维网包含数以亿计的 Web 页面，网络搜索引擎可能返回数以万计的页面。可以使用聚类将搜索结果分成若干簇，每个簇捕获查询的某个特定方面。例如，查询"电影"返回的网页可以分成诸如评价、电影预告片、影星和电影院等类别。每一个类别（簇）又可以划分成若干子类别（子簇），从而产生一个层次结构，帮助用户进一步探索查询结果。
- **心理学和医学**。一种疾病或健康状况通常有多种变种，聚类分析可以用来发现这些子类别。例如，聚类已经用来识别不同类型的抑郁症。聚类分析也可用来检测疾病的时间和空间分布模式。
- **商业**。商业点收集当前和潜在顾客的大量信息。可以使用聚类将顾客划分成若干组，以便进一步分析和开展营销活动。

除此之外，聚类作为一种数据挖掘工具还广泛应用于其他许多领域，如生物学、气候学、商务智能和 Web 搜索。

本章介绍聚类分析的基本概念和方法。在 6.1 节，引入该主题并介绍各种聚类方法和各种应用的要求。在 6.2 节至 6.5 节学习一些基本的聚类技术。在 6.6 节，简要讨论如何评估聚类方法。6.7 节给出一个案例分析以帮助读者更好地理解聚类分析方法。

6.1 基本概念和术语

本节给出聚类分析的基本概念及应用术语，为读者研究聚类分析建立基础。在 6.1.1 节，给出聚类分析的定义并给出一些例子。在 6.1.2 节，介绍聚类的基本要求。6.1.3 节给

出基本聚类方法的概述。

6.1.1 聚类分析简介

聚类分析(cluster analysis)简称**聚类**(clustering),是一个把数据对象(或观测)划分成子集的过程,每个子集是一个**簇**(cluster),使得簇中的对象彼此相似,但与其他簇中的对象不相似。组内的相似性越大,组间差别越大,聚类就越好。

聚类分析与其他将数据对象分组的技术相关。例如,聚类可以看作一种分类,它用类(簇)标号创建对象的标记。然而,只能从数据导出这些标号。相比之下,第5章的分类是**监督分类**(supervised classification),即使用由类标号已知的对象开发的模型对新的、无标记的对象赋予类标号。为此,有时称聚类分析为**非监督分类**(unsupervised classification)。

聚类分析已经广泛地应用于许多应用领域,包括商务智能、图像模式识别、Web搜索、生物学和安全。在商务智能应用中,聚类可以用来把大量客户分组,其中组内的客户具有非常类似的特征,这有利于开发加强客户关系管理的商务策略。此外,考虑具有大量项目的咨询公司,为了改善项目管理,可以基于相似性把项目划分成类别,使得项目审计和诊断(改善项目提交和结果)可以更有效地实施。

在图像识别应用中,聚类可以在手写字符识别系统中用来发现簇或“子类”。假设有手写数字的数据集,其中每个数字标记为1、2、3等。注意,人们写相同的数字可能存在很大差别。例如,数字“2”,有些人写的时候可能在左下方带一个小圆圈,而另一些人则不会。因此可以使用聚类确定“2”的子类,每个子类代表了手写数字“2”可能出现的变体。使用基于子类的多个模型可以提高整体识别的准确率。

在Web搜索中也有许多聚类应用。例如,由于Web网页的数量巨大,关键词搜索常常会返回大量命中对象(即与搜索相关的网页)。可以用聚类将搜索结果分组,以简明、容易访问的方式提交这些结果。此外,目前已经开发出把文档聚类成主题的聚类技术,这些技术已经广泛地用在实际的信息检索中。

作为一种数据挖掘功能,聚类分析也可以作为一种独立的工具,用来洞察数据的分布,观察每个簇的特征,将进一步分析集中在特定的簇集合上。另外,聚类分析可以作为其他算法(如特征化、属性子集选择和分类)的预处理步骤,之后这些算法将在检测到的簇和选择的属性或特征上进行操作。

6.1.2 对聚类的基本要求

聚类作为一种数据挖掘工具是一个富有挑战性的研究领域。本节学习对聚类的要求。

- **可伸缩性**。许多聚类算法在小于200个数据对象的小数据集合上工作得很好;但是,一个大规模数据库可能包含几百万个对象,在这样的大数据集合上进行聚类可能会导致有偏的结果。因此,需要具有高度可伸缩的聚类算法。
- **处理不同类型数据的能力**。许多算法被设计用来聚类数值类型的数据。但是,某些应用可能要求聚类其他类型的数据,如二元类型(binary)、分类/标称类型(categorical/nominal)、序数型(ordinal)数据,或者这些数据类型的混合。

- **发现任意形状的聚类**。许多聚类算法基于欧几里得或者曼哈顿距离度量来决定聚类。基于这样的距离度量的算法趋向于发现具有相近尺度和密度的球状簇。但是,一个簇可能是任意形状。重要的是开发能够发现任意形状的簇的算法。
- **用于决定输入参数的领域知识最小化**。许多聚类算法在聚类分析中要求用户输入一定的参数,例如希望产生的簇的数目。数据结果对输入参数十分敏感,参数通常很难确定,特别是对于包含高维对象的数据集来说。这样不仅加重了用户的负担,也使得聚类的质量难以控制。
- **处理"噪声"数据的能力**。绝大多数现实中的数据库都包含了孤立点、缺失或错误的数据。一些聚类算法对于这样的数据敏感,可能导致低质量的聚类结果。因此,需要对噪声鲁棒的聚类算法。
- **对于输入记录的顺序不敏感**。一些聚类算法对于输入数据的顺序是敏感的。例如,同一个数据集合,当以不同的顺序交给同一个算法时,可能生成差别很大的聚类结果。需要开发对数据输入顺序不敏感的算法。
- **聚类高维数据的能力**。数据集可能包含大量的维或属性。例如,在文档聚类时,每个关键词都可以看作一个维,并且常常有数以千计的关键词。许多聚类算法擅长处理低维数据,如只涉及两三个维的数据。发现高维空间中数据对象的簇是一个挑战,特别是在这样的数据可能非常稀疏,并且高度倾斜的情况下。
- **基于约束的聚类**。现实世界的应用可能需要在各种约束条件下进行聚类。假设你的工作是在一个城市中为给定数目的自动提款机选择安放位置。为了做出决定,你可以对住宅进行聚类,同时考虑城市的河流和公路网、每个簇(地区)的客户的类型和数量等情况。找到既满足特定的约束又具有良好聚类特性的数据分组是一项具有挑战性的任务。
- **可解释性和可用性**。用户希望聚类结果是可解释的、可理解的和可用的。也就是说,聚类可能需要与特定的语义解释和应用相联系。重要的是研究应用目标如何影响聚类特征和聚类方法的选择。

6.1.3 聚类分析方法

在如今的机器学习领域存在着大量的聚类算法,本节对各种不同的聚类方法提供一个相对有组织的描述以区别不同类型的聚类。

1. 划分方法

给定一个 n 个对象的集合,划分方法(partitioning method)构建数据的 k 个分组,其中每个分组表示一个簇,并且 $k \leqslant n$。而且这 k 个分组满足下列条件:

(1) 每一个分组至少包含一个数据记录。

(2) 每一个数据记录属于且仅属于一个分组(注意:这个要求在某些模糊聚类算法(6.5 节)中可以放宽)。

大部分划分方法是基于距离的。对于给定的分组数 k,算法首先给出一个初始的分组方法,以后通过反复迭代的方法改变分组,使得每一次改进之后的分组方案都较前一次

好,而所谓好的标准就是:同一分组中的记录越近越好,而不同分组中的记录越远越好。使用这个基本思想的算法有 k 均值算法、k 中心点算法、CLARANS 算法等。

为了达到全局最优,基于划分的聚类可能需要穷举所有可能的划分,计算量极大。实际上,大多数应用都采用了流行的启发式方法,如 k 均值和 k 中心点算法,渐进地提高聚类质量,逼近局部最优解。这些启发式聚类方法很适合发现中小规模的数据库中的球状簇。为了发现具有复杂形状的簇和对超大型数据集进行聚类,需要进一步扩展基于划分的方法。在 6.2 节,将深入研究基于划分的聚类方法。

2. 层次方法

层次方法(hierarchical method)给出了给定数据对象集的层次分解。根据层次分解如何形成,层次方法可以分为**凝聚**的或**分裂**的方法。**凝聚的方法**也称**自底向上**的方法,开始将每个对象作为单独的一个组,然后逐次合并相近的对象或组,直到所有的对象合并为一个组(层次的最顶层),或者满足某个终止条件。**分裂的方法**也称为**自顶向下**的方法,开始将所有的对象置于一个簇中。在每次相继迭代中,一个簇被划分成更小的簇,直到最终每个对象在单独的一个簇中,或者满足某个终止条件。层次方法的代表算法有 BIRCH 算法、CURE 算法、Chameleon 算法等。

层次聚类方法可以是基于距离的或基于密度或连通性的。层次方法的缺陷在于,一旦一个步骤(合并或分裂)完成,它就不能被撤销。这个严格规定是有用的,因为担心不同选择的组合数目,它将产生较小的计算开销。然而,这种技术不能更正错误的决定。目前已经提出了一些提高层次聚类质量的方法。层次聚类方法将在 6.3 节介绍。

3. 基于密度的方法

大部分划分方法基于对象之间的距离进行聚类。这样的方法只能发现球状簇,而在发现任意形状的簇时会遇到困难。已经开发了基于密度概念的聚类方法(density-based method),其主要思想是:只要一个区域中的点的密度(对象或数据点的数目)超过某个阈值,就把它加到与之相近的聚类中去。也就是说,对给定簇中的每个数据点,在给定半径的邻域中必须至少包含最少数目的点。这样的方法可以用来过滤噪声或离群点,发现任意形状的簇。代表算法有 DBSCAN 算法、OPTICS 算法、DENCLUE 算法等。

基于密度的方法可以把一个对象集划分成多个互斥的簇或簇的分层结构。通常,基于密度的方法只考虑互斥的簇,而不考虑模糊簇。此外,可以把基于密度的方法从整个空间聚类扩展到子空间聚类。基于密度的聚类方法在 6.4 节介绍。

4. 基于网格的方法

基于网格的方法(grid-based method)把对象空间量化为有限个单元(cell),形成一个网格结构,所有的处理都是以单个的单元为对象的。这种方法的主要优点是处理速度很快,通常这是与目标数据库中记录的个数无关的,它只与把数据空间分为多少个单元有关。代表算法有 STING 算法、CLIQUE 算法、WAVE-CLUSTER 算法。

5. 基于模型的方法

基于模型的方法(model-based method)给每一个聚类假定一个模型,然后寻找能够很好地满足这个模型的数据集。这样一个模型可能是数据点在空间中的密度分布函数等。它的一个潜在的假定就是:目标数据集是由一系列的概率分布所决定的。这种方法通常有两种尝试方向:统计的方案和神经网络的方案。

表 6-1 简略地总结了这些方法。有些聚类方法集成了多种聚类方法的思想,因此有时很难将一个给定的算法只划归到一个聚类方法类别。此外,有些应用可能有某种聚类准则,要求集成多种聚类技术。

表 6-1　本章讨论的聚类方法概述

方　法	一般特点
划分方法	• 发现球状互斥的簇 • 基于距离 • 可以用均值或中心点等代表簇中心 • 对中小规模数据集有效
层次方法	• 聚类是一个层次分解(即多层) • 不能纠正错误的合并或划分 • 可以集成其他技术,如微聚类或考虑对象“连接”
基于密度的方法	• 可以发现任意形状的簇 • 簇是对象空间中被低密度区域分隔的稠密区域 • 簇密度规定了每个点的“邻域”内必须具有最少个数的点 • 可以过滤离群点
基于网格的方法	• 使用一种多分辨率网格数据结构 • 快速处理(典型地,独立于数据对象数,但依赖于网格大小)
基于模型的方法	• 假设模型寻找满足的数据集 • 不适合对大数据库进行聚类

6.2　基于划分的方法

聚类分析最简单、最基本的方法是划分,它把对象组织成多个互斥的组或簇。在这里,为了使得问题的描述更简洁,假定簇的个数已给定。

给定 n 个数据对象的数据集 D 以及要生成的簇数 k。**划分方法**将 D 中的对象分配到 k 个簇 $C_1, C_2, \cdots, C_k$ 中,一个目标函数用来评估划分的质量,使得簇内对象相似,而与其他簇中的对象相异。也就是说,该目标函数以簇内高相似性和簇间低相似性为目标。

本节学习最常用的划分方法——k-means(6.2.1 节)和 k 中心点(6.2.2 节)。

6.2.1　*k*-means 算法

k-means(k 均值)是一种基于形心的划分技术,即使用簇 C_i 的形心代表该簇。k-means 算法把簇的形心定义为簇内点的均值,它的算法比较简单,首先介绍它的处理流程。

首先，在数据集 D 中随机地选择 k 个对象，每个对象代表一个簇的初始均值或中心。对剩下的每个对象，根据其与各个簇中心的欧氏距离，把它分配到最相似的簇。然后，对于每个簇，使用上次迭代分配到该簇的对象，计算新的均值。然后，使用更新后的均值作为新的簇中心，重新分配所有对象。继续迭代，直到分配稳定，即本轮形成的簇与前一轮形成的簇相同。k-means 的形式描述在算法 6-1 中。

算法 6-1 用于划分的 k-means 算法

输入：

k：簇的数目；

D：包含 n 个对象的数据集。

输出：k 个簇的集合

步骤：

(1) 从 D 中任意选择 k 个对象作为初始簇中心；

(2) repeat

(3) 根据簇中对象的均值，将每个对象分配到最相似的簇；

(4) 更新簇均值，即重新计算每个簇中对象的均值；

(5) until 不再发生变化。

接下来详细地考虑基本 k-means 算法的每个步骤，并给出一个实例的计算过程。

1. 分配到最近的质心

为了将数据点分配到最近的质心（中心），需要邻近性度量来量化所考虑的数据的“最近”概念。通常，对欧氏空间中的点使用欧几里得距离，对文档用余弦相似性。然而，对于给定的数据类型，可能存在多种适合的邻近性度量。

通常，k-means 使用的相似性度量相对简单，因为算法要重复地计算每个点与每个质心的相似度。然而，在某些情况下，如数据在低维欧几里得空间时，许多相似度的计算都有可能避免，因此显著地加快了 k-means 算法的速度。二分 k-means 就是一种通过减少相似度计算量来加快算法速度的方法。

2. 质心和目标函数

k-means 算法步骤(4)一般陈述为“重新计算每个簇的质心”，因为质心可能随数据邻近性度量和聚类目标不同而改变。聚类的目标通常用一个目标函数表示，该函数依赖于点之间或点到簇的质心的邻近性，如最小化每个点到最近质心的距离的平方。

考虑邻近性度量为欧几里得距离的数据，使用**误差的平方和**(Sum of the Squared Error，SSE)作为度量聚类质量的目标函数。换言之，计算每个数据点的误差，即它到最近质心的欧几里得距离，然后计算误差的平方和。给定由两次运行 k-means 算法产生的两个不同的簇集，我们更喜欢误差的平方和最小的那个，因为这说明此时聚类的质心可以更好地代表簇的中点。SSE 的定义如下：

$$\text{SSE} = \sum_{i=1}^{k} \sum_{x \in C_i} \text{dist}\,(c_i, x)^2 \tag{6-1}$$

其中，dist 是标准欧几里得距离，k 是类簇个数，x 是类簇 C_i 中的一个数据对象，c_i 是类簇 C_i 质心。

步骤(3)和步骤(4)试图直接最小化 SSE(或更一般地，目标函数)。步骤(3)通过将对象分配到最近的质心形成簇，最小化关于给定质心集的 SSE；而步骤(4)重新计算质心，进一步最小化 SSE。然而，k-means 的步骤(3)和步骤(4)只能确保找到关于 SSE 的局部最优，因为它们是对选定的质心和簇而不是对所有可能的选择来优化 SSE。

3. 选择初始质心

选择适当的初始质心是基本 k-means 过程的关键步骤。常见的方法是随机地选取初始质心，但是簇的质量常常很差。实践中，为了得到好的结果，通常以不同的初始簇中心多次运行 k-means 算法。

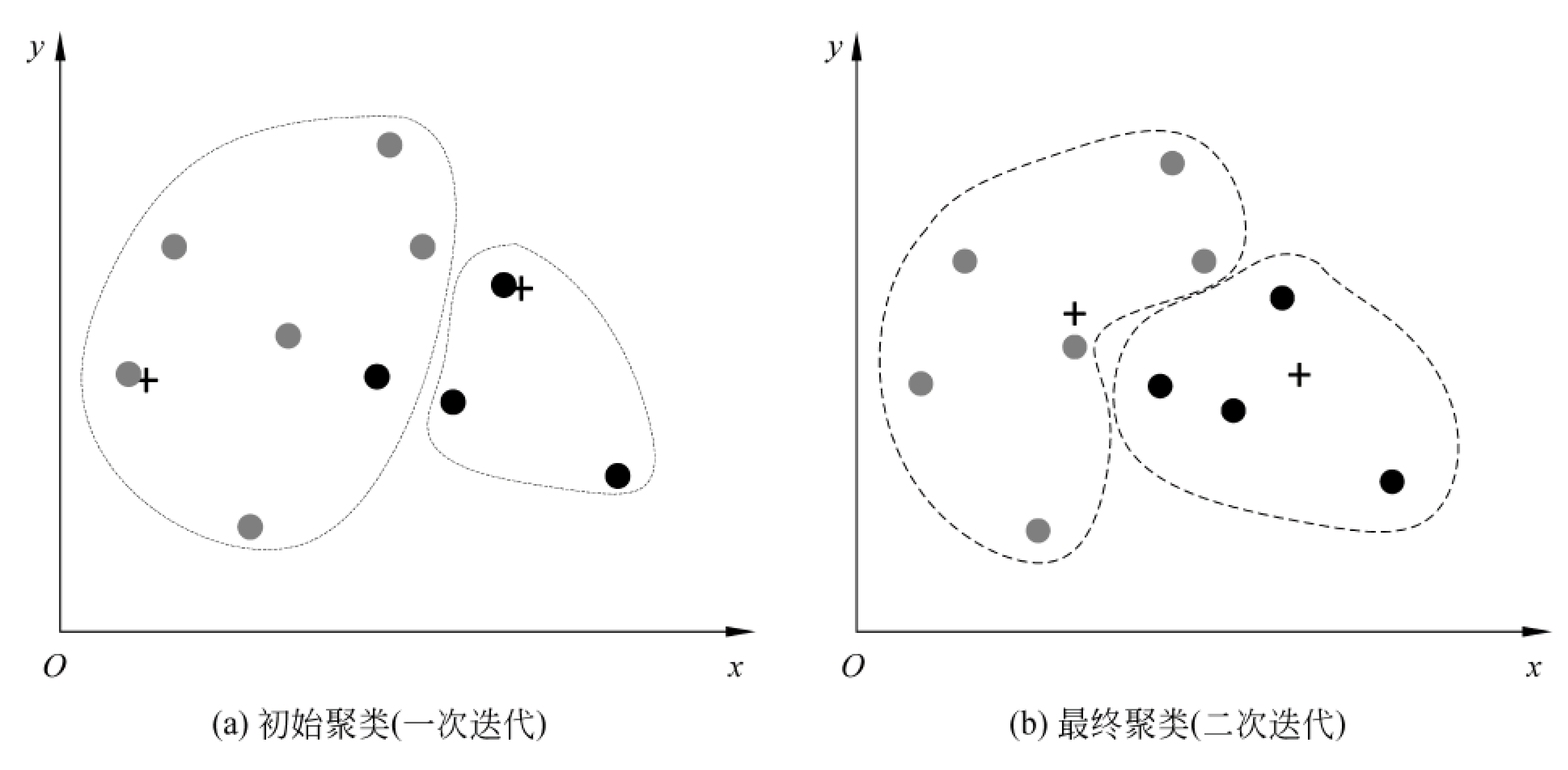
例 6-1 使用 k-means 的聚类划分。

考虑二维空间中的 10 个点，它们的值分别为 $A_1(0.5,2)$，$A_2(0.8,3)$，$A_3(1.2,0.6)$，$A_4(1.6,2.2)$，$A_5(2.4,3.6)$，$A_6(2.5,2.8)$，$B_1(2.2,1.8)$，$B_2(3,2.5)$$B_3(2.8,1.6)$，$B_4(4,1)$，如图 6-1(a)所示，其中，$A_1$、$A_2$、$A_3$、$A_4$、$A_5$、$A_6$ 用灰色实心圆表示，B_1、B_2、B_3、B_4 用黑色实心圆表示。令 $k=2$，即用户要求将这些对象划分成两个簇。

根据算法 6-1，任意选择两个对象作为两个初始的簇中心，其中簇中心用"+"标记。

(1) 假设选择 B_1、B_3 分别为每个簇的初始中心，即 $O_1=B_1=(2.2,1.8)$，$O_2=B_3=(2.8,1.6)$。

(2) 对剩余的每个对象，根据其与各个簇中心的距离，将它赋给最近的簇。这种分配形成了如图 6-1(a)中虚线所描绘的轮廓。



(a) 初始聚类(一次迭代)　　(b) 最终聚类(二次迭代)

图 6-1　使用 k-means 方法聚类对象集

对于 A_1，它与两个簇中心的距离为

$$d(O_1,A_1)=\sqrt{(0.5-2.2)^2+(2-1.8)^2}=1.712$$

$$d(O_2,A_1)=\sqrt{(0.5-2.8)^2+(2-1.6)^2}=2.377$$

显然 $d(O_1,A_1)<d(O_2,A_1)$，故将 A_1 分配给 O_1 所在簇。

同样地，对于 B_2，它与两个簇中心的距离为

$$d(O_1,B_2)=\sqrt{(3-2.2)^2+(2.5-1.8)^2}=1.063$$

$$d(O_2,B_2)=\sqrt{(3-2.8)^2+(2.5-1.6)^2}=0.912$$

这里 $d(O_1,B_2)>d(O_2,B_2)$，故将 B_2 分配给 O_2 所在簇。

对剩余对象都分配完后，得到的新簇为 $O_1=\{A_1,A_2,A_3,A_4,A_5,A_6,B_1\}$，$O_2=\{B_2,B_3,B_4\}$，这样就形成了如图 6-1(a)中虚线所描绘的轮廓。

(3) 计算新的簇的中心：

$$O_1=((0.5+0.8+1.2+1.6+2.2+2.4+2.5))/7,$$
$$((2+3+0.5+2.2+1.3+3.6+3.8)/7)=(1.6,2.3)$$
$$O_2=((2.8+3+4)/3,(1.5+2.5+1)/3)=(3.3,1.7)$$

重复步骤(2)，更新得到新簇 $O_1=\{A_1,A_2,A_3,A_4,A_5,A_6\}$，$O_2=\{B_1,B_2,B_3,B_4\}$，形成如图 6-1(b)中虚线所描绘的轮廓。

重复以上过程，在第三次迭代之后，此时簇中心不再改变，停止迭代过程，算法停止，最终划分结果如图 6-1(b)所示。

k-means 算法的复杂度为 $O(nkt)$，其中 n 是对象总数，k 是簇数，t 是迭代次数。通常 $k\ll n$ 并且 $t\ll n$。因此，对于处理大数据集，该算法是相对可伸缩的和有效的。

k-means 方法有一些变种，它们可能在初始 k 个均值的选择、相异度的计算、簇均值的计算策略上有所不同。然而，对于发现不同的簇类型，*k*-means 和它的变种都具有一些局限性。具体地说，当簇具有非球形形状或具有不同尺寸或密度时，*k*-means 很难检测到"自然的"簇。如图 6-2、图 6-3 和图 6-4 所示。在图 6-2 中，*k*-means 不能发现那 3 个自然簇，因为其中一个簇比其他两个大得多，因此较大的簇被划分开，而一个较小的簇与较大簇的一部分合并在一起。在图 6-3 中，*k*-means 未能发现那 3 个自然簇，因为两个较小的簇比较大的簇稠密得多。在图 6-4 中，*k*-means 发现了两个簇(两个自然簇的混合体)，因为两个自然簇的形状不是球形的。

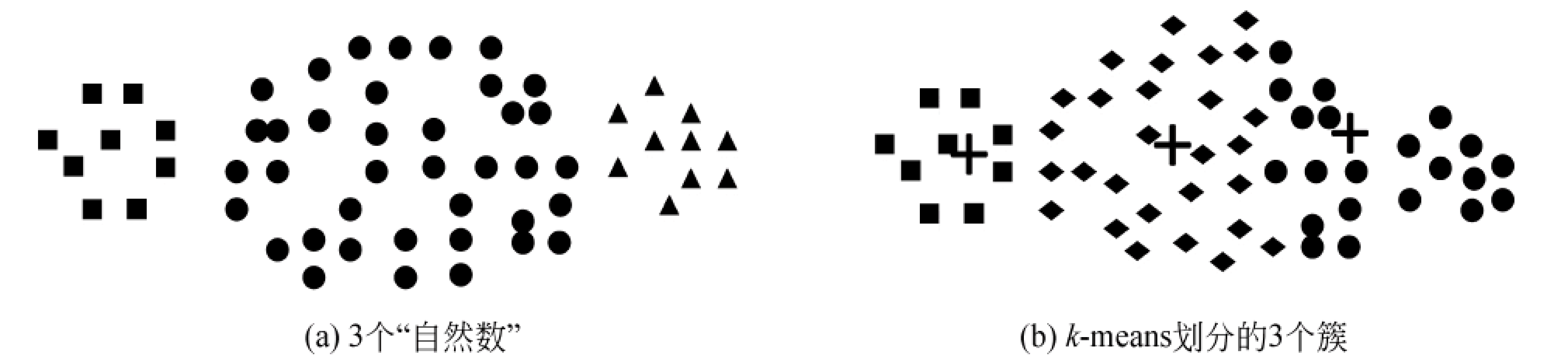


图 6-2 *k*-means：具有不同尺寸的簇

这 3 种情况的问题在于 *k*-means 的目标函数与我们试图发现的簇的类型不匹配，因为 *k*-means 目标函数是最小化等尺寸和等密度的球形簇或者明显分离的簇。

6.2.2 *k* 中心点算法

k-means 算法对离群点敏感，因为这种对象远离大多数数据，因此分配到一个簇时，

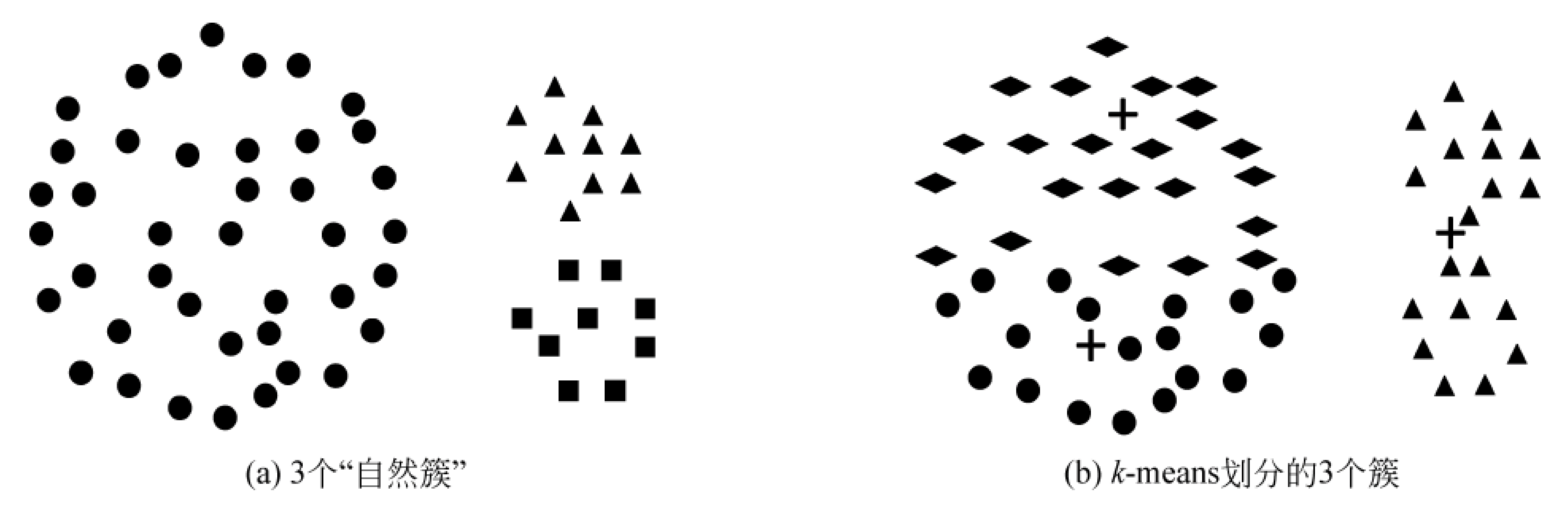


图 6-3 k-means：具有不同密度的簇

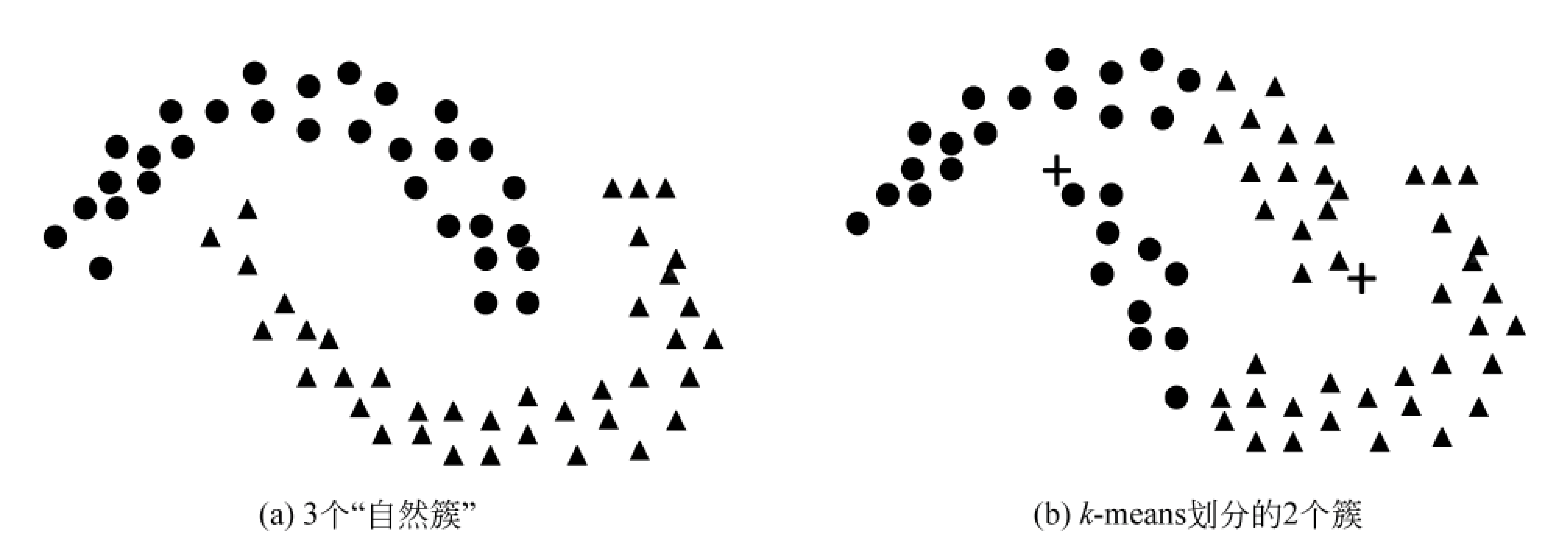


图 6-4 k-means：非球形的簇

它们可能严重地扭曲了簇的均值，这不经意间影响了其他对象到簇的分配。正如在例 6-2 中所观察到的，式(6-1)平方误差函数的使用更是严重恶化了这一影响。

例 6-2 k-means 的缺点。

考虑一维空间的 7 个点，它们的值分别为 1，2，3，8，9，10，25。

直观地，通过视觉观察，我们猜想这些点划分成簇{1，2，3}和{8，9，10}，而点 25 被排除，因为它看上去是一个离群点。k-means 如何划分这些值？如果以 $k=2$ 和式(6-1)使用 k-means 划分为{{1，2，3}，{8，9，10，25}}，具有簇内变差

$$(1-2)^2+(2-2)^2+(3-2)^2+(8-13)^2+(9-13)^2+(10-13)^2 \\ +(25-13)^2 \\ =196$$

其中，簇{1，2，3}的均值为 2，簇{8，9，10，25}的均值为 13。把这一划分与划分{{1，2，3，8}，{9，10，25}}比较，后者的簇内变差为

$$(1-3.5)+(2-3.5)^2+(3-3.5)^2+(8-3.5)^2+(9-14.67)^2 \\ +(10-14.67)^2+(25-14.67)^2 \\ =189.67$$

其中，簇{1，2，3，8}的均值为 3.5，簇{9，10，25}的均值为 14.67。后一个划分具有最小簇内变差。因此，由于离群点 25 的缘故，k-means 方法没有把 8 分配到 9 和 10 所在的簇。此外，第二个簇中心为 14.67，显著地偏离簇中的所有成员。

如何修改 k-means 算法，降低它对离群点的敏感性？可以不采用簇中对象的均值作为参照点，而是挑选实际对象来代表簇，每个簇使用一个代表对象，其余的每个对象被分配到与其最为相似的代表性对象所在的簇中。于是，划分方法基于最小化所有对象 p 与其对应的代表对象之间的相异度之和的原则来进行划分。确切地说，使用了一个**绝对误差标准**(absolute-error criterion)，其定义如下：

$$E=\sum_{i=1}^{k}\sum_{p\in C_j}\text{dist}(p,o_i) \tag{6-2}$$

其中，E 是数据集中所有对象 p 与类 C_i 的代表对象 o_i 的绝对误差之和。这是 k 中心点(k-medoids)方法的基础。k 中心点聚类通过最小化该绝对误差(式(6-2))，把 n 个对象划分到 k 个簇中。

围绕中心点划分(Paritioning Around Mediods，PAM)算法(算法 6-2)是 k 中心点聚类的一种流行实现。它用迭代、贪心的方法处理该问题。与 k-means 算法一样，初始代表对象(称作种子)任意选取。考虑用一个非代表对象替换一个代表对象是否能够提高聚类质量。尝试所有可能的替换。继续用其他的对象替换代表对象的迭代过程，直到结果聚类的质量不可能被任何替换提高。聚类质量用对象与其簇中代表对象的平均相异度的代价函数度量。

算法 6-2 PAM，一种基于中心的 k 中心点算法

输入：

k：簇的数目

D：包含 n 个对象的数据集

输出：k 个簇的集合

步骤：

(1) 从 D 中任意选择 k 个对象作为初始的代表对象或种子；

(2) repeat

(3) 将每个剩余的对象分配到最近的代表对象所代表的簇；

(4) 随机地选择一个非代表对象 O_{random}；

(5) 计算用 O_{random} 代替代表对象 O_j 的总代价 S；

(6) if $S<0$，then O_{random} 替换 O_j，形成新的 k 个代表对象的集合；

(7) until 不再发生变化。

具体地说，设 $O_1,O_2,\cdots,O_k$ 是当前代表对象(即中心点)的集合。为了决定一个非代表对象 O_{random} 是否是一个当前中心点 $O_j(1\leqslant j\leqslant k)$ 的好的替代，计算每个对象 p 到集合 $\{O_1,\cdots,O_{j-1},O_{\text{random}},O_{j+1},\cdots,O_k\}$ 中最近对象的距离，并使用该距离更新代价函数。对象重新分配到 $\{O_1,\cdots,O_{j-1},O_{\text{random}},O_{j+2},\cdots,O_k\}$ 中是简单的。假设对象 p 当前被分配到中心点 O_j 代表的簇中(见图 6-5(a)或图 6-5(b))。在 O_j 被 O_{random} 置换后，需要把 p 重新分配到不同的簇吗？对象 p 需要重新分配，被分配到 O_{random} 还是其他 $O_i(i\neq j)$ 代表的簇，取决于哪个最近。例如，在图 6-5(a)中，p 离 O_i 最近，因此它被重新分配到 O_i。然而，在图 6-5(b)中，p 离 O_{random} 最近，因此它被重新分配到 O_{random}。如果 p 当前被分配到其他对象 $O_i(i\neq j)$ 代表的簇中又该怎么办？只要对象 p 离 O_i 仍然比离 O_{random} 更近，那么它就仍

然被分配到 O_i 代表的簇(见图 6-5(c))。否则,p 被重新分配到 O_{random}(见图 6-5(d))。图 6-5 中,实箭线表示替换前的分配情况,虚箭线表示替换后的分配情况。

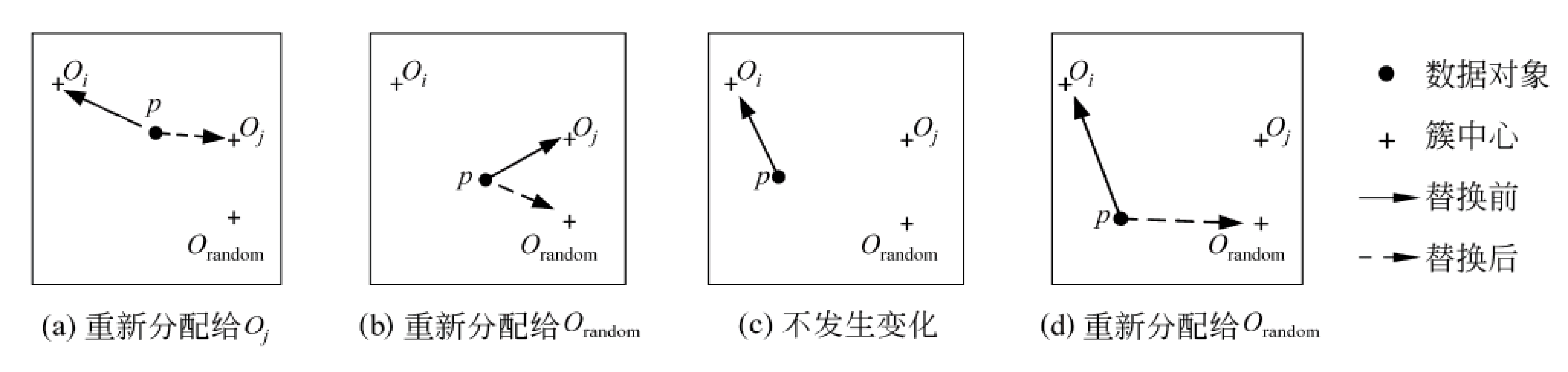


图 6-5 k 中心点聚类代价函数的 4 种情况

每当重新分配发生时,绝对误差 E 的差对代价函数有影响。因此,如果一个当前的代表对象被非代表对象所取代,则代价函数就计算绝对误差值的差。交换的总代价是所有非代表对象所产生的代价之和。如果总代价为负,则实际的绝对误差 E 将会减小,O_j 可以被 O_{random} 取代或交换;如果总代价为正,则认为当前的代表 O_j 是可接受的,在本次迭代中没有发生变化。

当存在噪声和离群点时,k 中心点方法比 k-means 更鲁棒,这是因为中心点不像均值那样容易受离群点或其他极端值影响。然而,k 中心点每次迭代的复杂度是 $O(k(n-k))$。当 n 和 k 的值很大时,这种计算开销远高于 k-means 方法。另外,这两种方法都要求用户指定簇数 k。

像 PAM(算法 6-2)这样的典型的 k 中心点算法在小型数据集上运行良好,但是不能很好地用于大数据集。为了处理大数据集,可以使用一种称作 CLARA(Clustering LARge Applications,聚类大型应用)的基于抽样的方法。CLARA 并不考虑整个数据集合,而是使用数据集的一个随机样本。最后使用 PAM 方法由样本计算最佳中心点。

6.3 基于层次的方法

在某些应用中,想把数据划分成不同层的组群,数据具有一个我们想要发现的基本层次结构。对于数据汇总和可视化,用层次结构的形式表示数据对象是有用的。例如,层次聚类可以揭示企业雇员在收入上的分层结构,可以把雇员组织成较大的组群,如主管、经理和职员。把这些组进一步划分成较小的子组群。例如,一般的职员组可以进一步划分成高级职工、职员和实习人员。所有这些组群形成了一个层级结构。在进化研究中,层次聚类可以按动物的生物学特征对它们分组,发现进化路径,即物种的分层结构。

本节学习层次聚类。在 6.3.1 节,首先介绍凝聚的和分裂的层次聚类。对层次聚类算法的距离度量将在 6.3.2 节展开。

6.3.1 凝聚的与分裂的层次聚类

层次聚类技术是第二类重要的聚类方法。和 k-means 一样,与许多聚类方法相比,层次聚类方法相对较老,但是它们仍然被广泛使用。根据层次分解的形成方式是自底向上

(合并)还是自顶向下(分裂),有两种产生层次聚类的基本方法。

凝聚的层次聚类方法使用自底向上的策略。从每个对象作为个体簇开始,每一步合并两个最接近的簇。这需要定义簇的邻近性概念。

分裂的层次聚类方法使用自顶向下的策略。从包含所有对象的某个簇开始,每一步分裂一个簇,直到仅剩下单点簇。在这种情况下,需要确定每一步分裂哪个簇,以及如何分裂。

在凝聚的或分裂的层次聚类中,用户都可以指定期望的簇个数作为终止条件。

例 6-3 凝聚和分裂层次聚类。

图 6-6 显示了一种凝聚的层次聚类算法和一种分裂的层次聚类算法在一个包含 5 个对象的数据集{a,b,c,d,e}上的处理过程。初始时,凝聚方法将每个对象自成一簇,然后这些簇根据某种准则逐步合并。例如,如果簇 C_1 中的一个对象和簇 C_2 中的一个对象之间的距离是所有属于不同簇的对象间欧氏距离中最小的,则 C_1 和 C_2 可能被合并。簇合并过程反复进行,直到所有对象最终合并到一个簇。

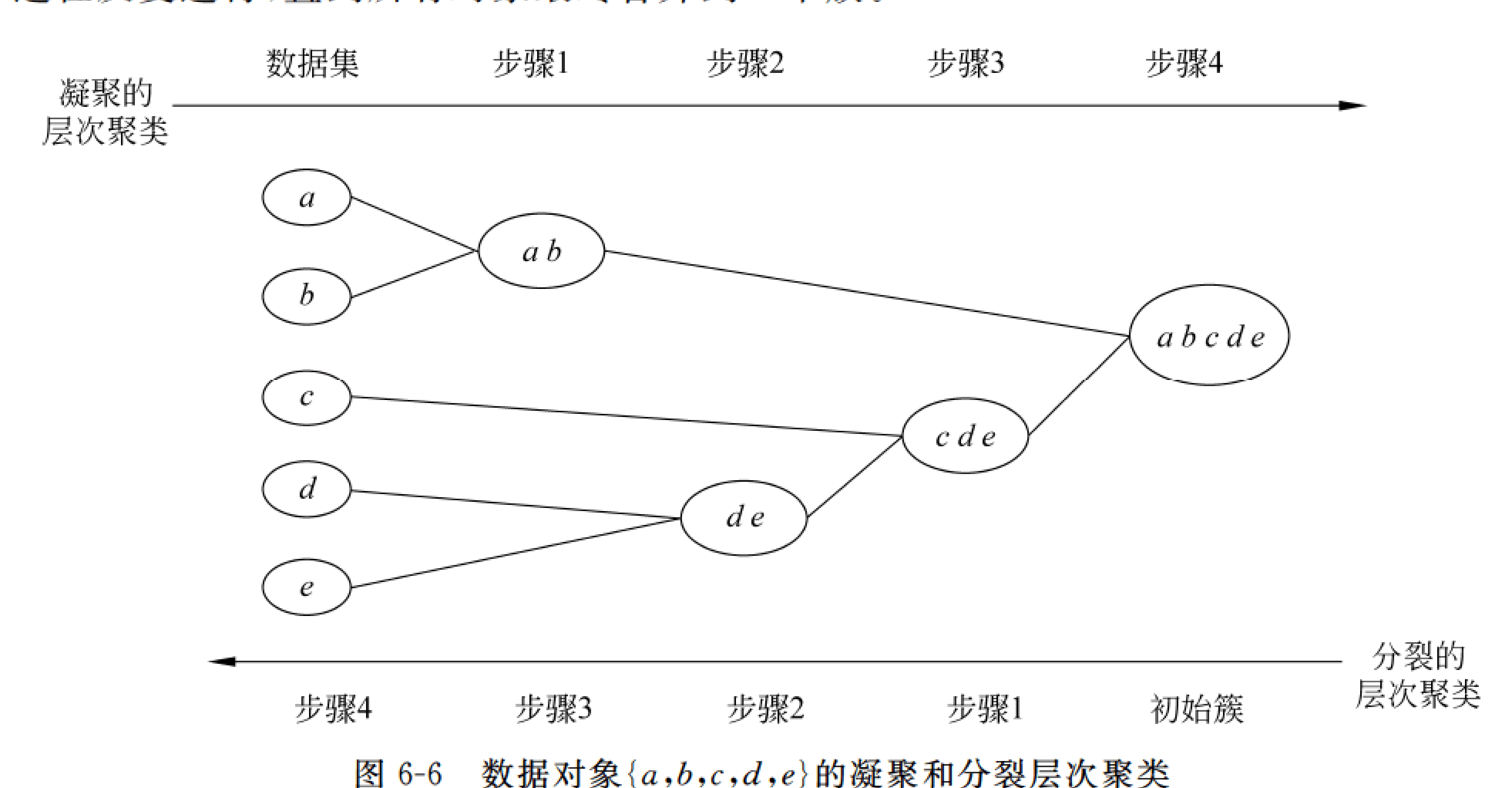


图 6-6 数据对象{a,b,c,d,e}的凝聚和分裂层次聚类

分裂方法以相反的方法处理。所有的对象形成一个初始簇,根据某种原则(如簇中最近的相邻对象的最大欧氏距离)将该簇分裂。簇的分裂过程反复进行,直到最终每个新的簇只包含一个对象。

层次聚类常常使用称作**树状图**(dendrogram)的类似于树的图显示。该图显示簇—子簇的联系和簇凝聚或分裂的次序。对于二维点的集合,层次聚类也可以使用**嵌套簇图**(nested cluster diagram)表示。图 6-7 是对例 6-3 中的 5 个对象的数据集进行层次聚类的结果,显示了这两种图的例子。

6.3.2 簇间距离度量

无论使用凝聚方法还是分裂方法,一个核心问题是度量两个簇之间的距离,其中每个簇一般是一个对象集。

根据簇间距离度量方法的不同,可将层次聚类分为**单连接**(single-linkage)、**全连接**

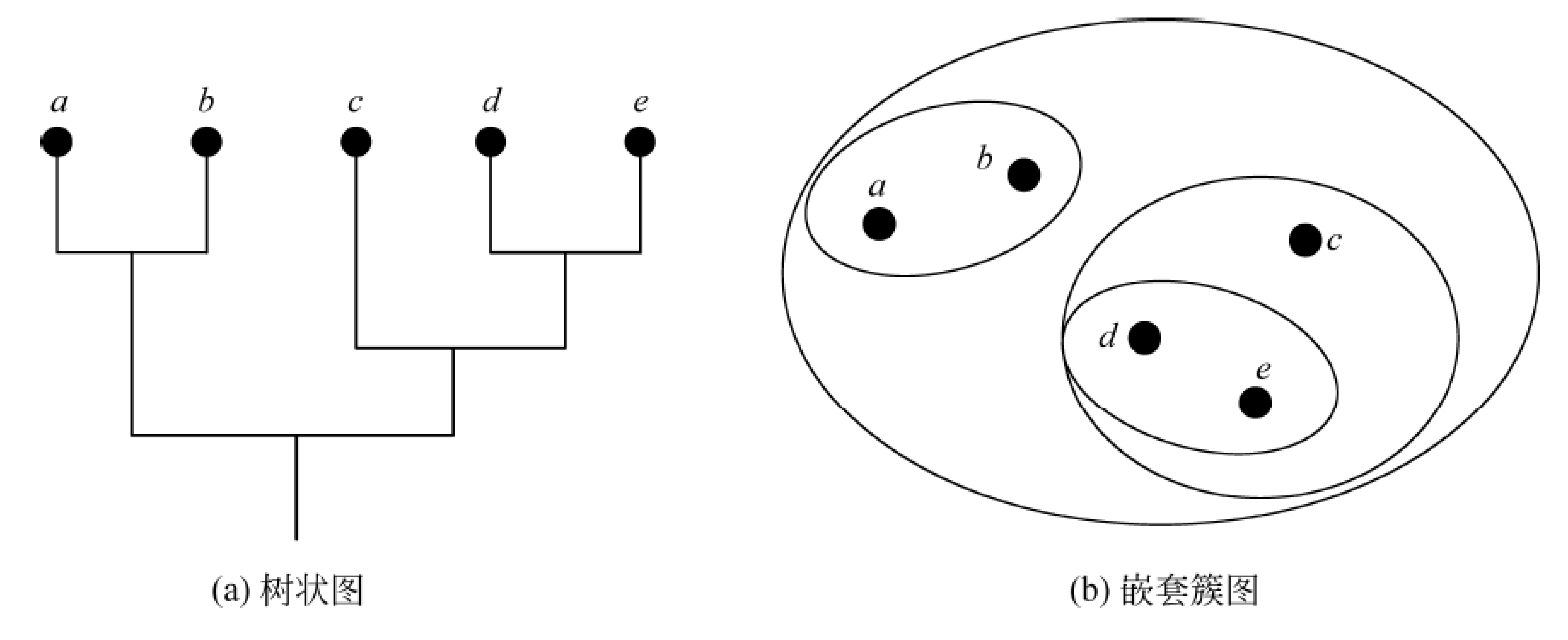


图 6-7　以树状图和嵌套簇图显示的 5 个对象的层次聚类

(complete-linkage)和**平均连接**(average-linkage)3 种方法，其中单连接采用的是最小距离，全连接采用的是最大距离，平均连接采用的是平均距离。3 种距离定义如下，其中 $|p-p'|$ 是两个对象之间的距离。

$$\text{最小距离：} \mathrm{dist}_{\min}(C_i,C_j)=\min_{p\in C_i,p'\in C_j}\{|p-p'|\} \tag{6-3}$$

$$\text{最大距离：} \mathrm{dist}_{\max}(C_i,C_j)=\max_{p\in C_i,p'\in C_j}\{|p-p'|\} \tag{6-4}$$

$$\text{平均距离：} \mathrm{dist}_{\mathrm{avg}}(C_i,C_j)=\frac{1}{n_i n_j}\sum_{p\in C_i,p'\in C_j}|p-p'| \tag{6-5}$$

这 3 种层次聚类技术都源于簇的基于图的观点。单连接(也称最小距离)定义簇间距离为不同簇的两个最近的点之间的距离，或者使用图的术语，是不同的节点子集中两个节点之间的最短边。全连接(也称最大距离)取不同簇中两个最远的点之间的距离作为簇间距离，或者使用图的术语，是不同的节点子集中两个节点之间的最长边。平均连接(也称平均距离)定义簇间距离为取自不同簇的所有点对距离的平均值。图 6-8 给出了这 3 种方法的示意图。

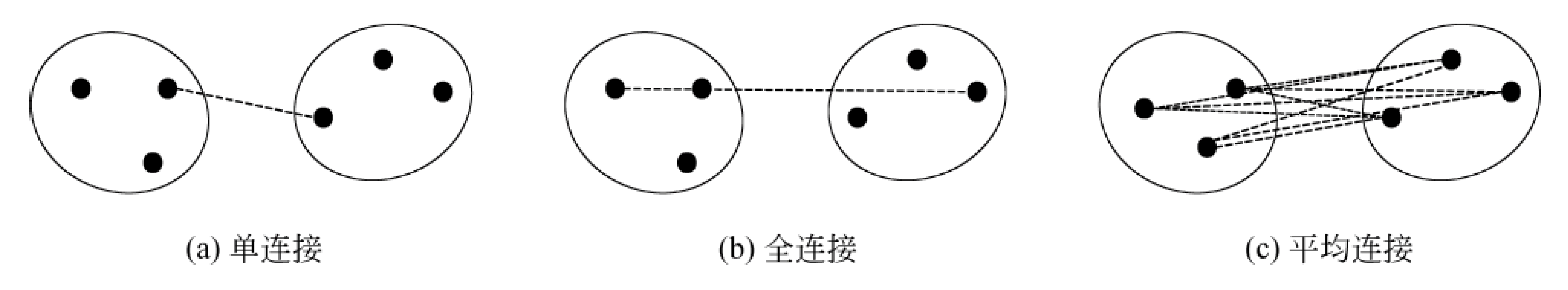


图 6-8　簇间距离度量基于图的定义

例 6-4　连接度量的不同使用。

为了解释不同距离度量方法，本例使用包含 6 个二维点的样本数据，如图 6-9 所示。数据点之间的欧几里得距离如图 6-10 所示。

(1) **单连接(最小距离)**。对于层次聚类的单连接或最小距离技术，两个簇的距离定义为两个簇中任意两点之间的最短距离(最大相似度)。使用图的术语，如果从所有点作为单点簇开始，每次在点之间加上一条链，最短的链先加，则这些链将合并成簇。单连接技术擅长处理非椭圆形状的簇，但对噪声和离群点很敏感。

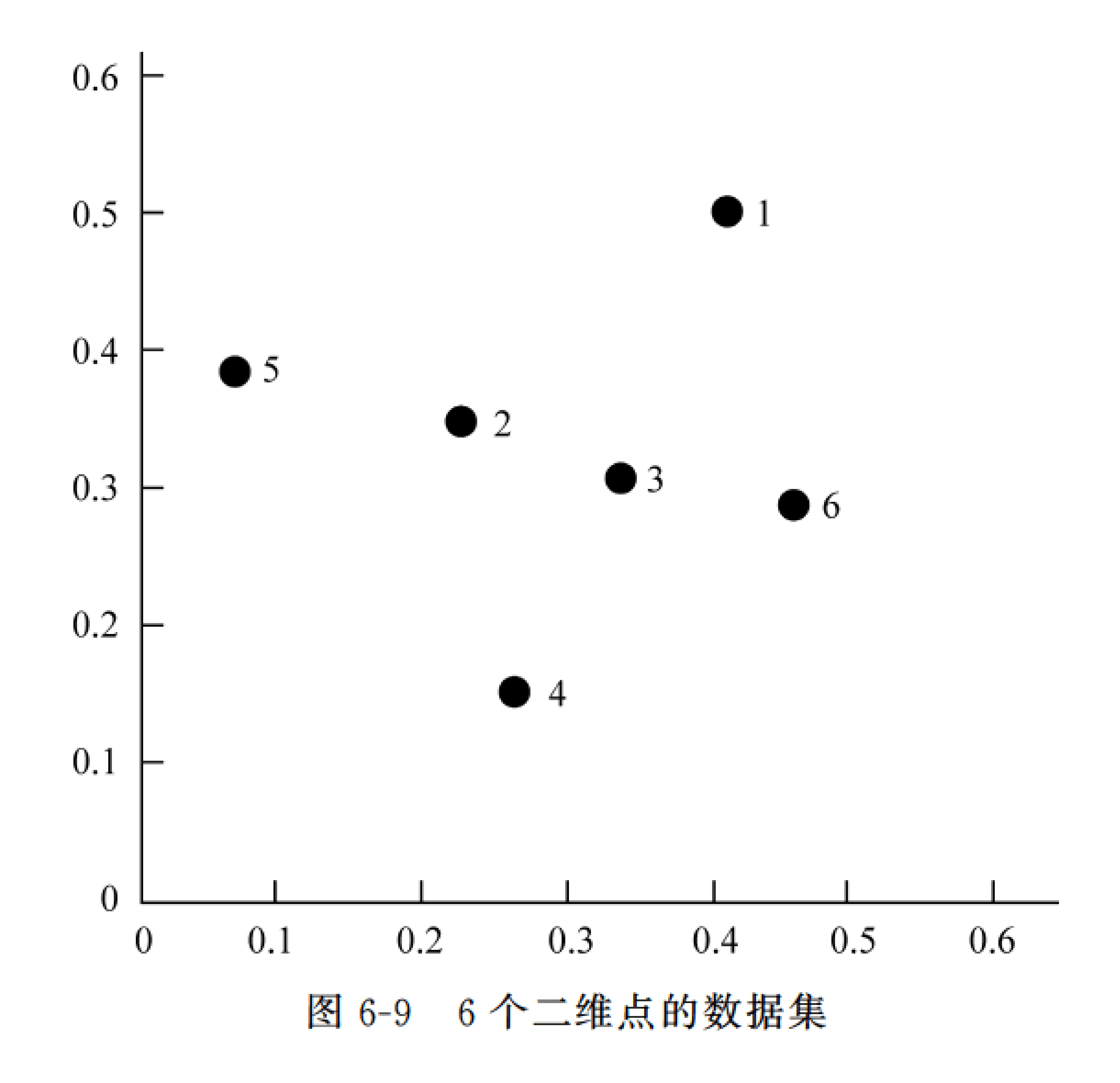


图 6-9　6 个二维点的数据集

	p1	p2	p3	p4	p5	p6
p1	0.0000	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0.0000	0.1483	0.2042	0.1388	0.2540
p3	0.2218	0.1483	0.0000	0.1513	0.2843	0.1100
p4	0.3688	0.2042	0.1513	0.0000	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0.0000	0.3921
p6	0.2347	0.2540	0.1100	0.2216	0.3921	0.0000

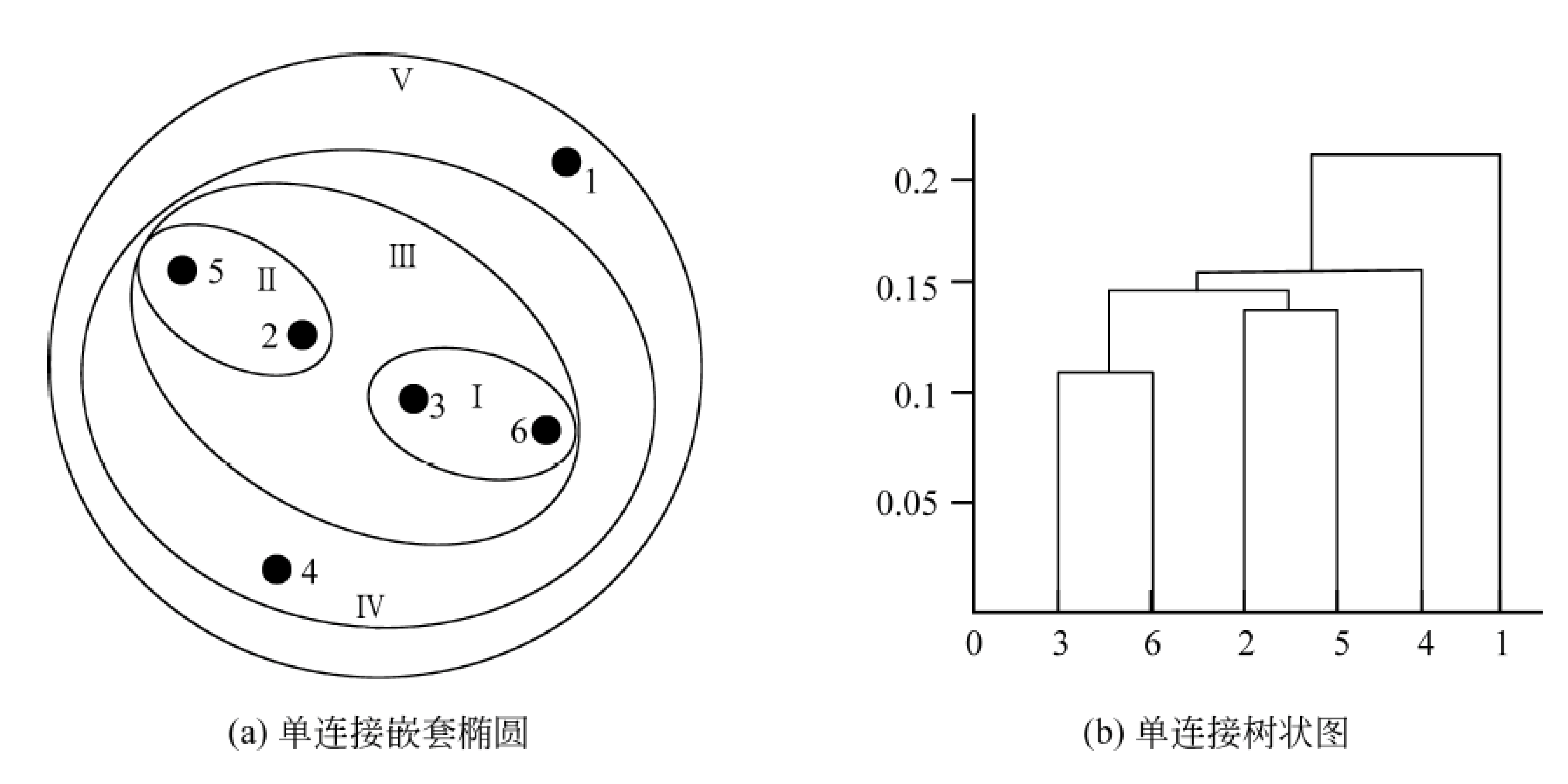
图 6-10　6 个点的欧几里得距离矩阵

图 6-11 显示了将单连接技术用于 6 个点数据集例子的结果。图 6-11(a)用嵌套的椭圆序列显示嵌套的簇，其中与椭圆相关联的数指示聚类的次序。图 6-11(b)显示了同样的信息，但使用树状图表示，树状图中两个簇合并处的高度反映两个簇的距离。例如，由图 6-10，我们看到点 3 和 6 的距离是 0.11，这就是它们在树状图里合并处的高度。再看另一个例子，簇{3,6}和{2,5}之间的距离是

$$\begin{aligned} dist(\{3,6\},\{2,5\}) &= \min(dist(3,2),dist(6,2),dist(3,5),dist(6,5)) \\ &= \min(0.15,0.25,0.28,0.29) \\ &= 0.15 \end{aligned}$$

(2) **全连接(最大距离)**。对于层次聚类的全连接或最大距离技术，两个簇的距离定义为两个不同簇中任意两点之间的最长距离(最小相似度)。使用图的术语，如果从所有点作为单点簇开始，每次在点之间加上一条链，最短的链先加，则一组点直到其中所有的点都完全被连接才形成一个簇。全连接对噪声和离群点不太敏感，但是它可能使得大的簇破裂，并且偏好球状。

图 6-12 显示了将全连接用于 6 个样本数点集的结果。与单连接一样，点 3 和 6 首先合并。然后，{3,6}与{4}合并，而不是与{2,5}或{1}合并，因为



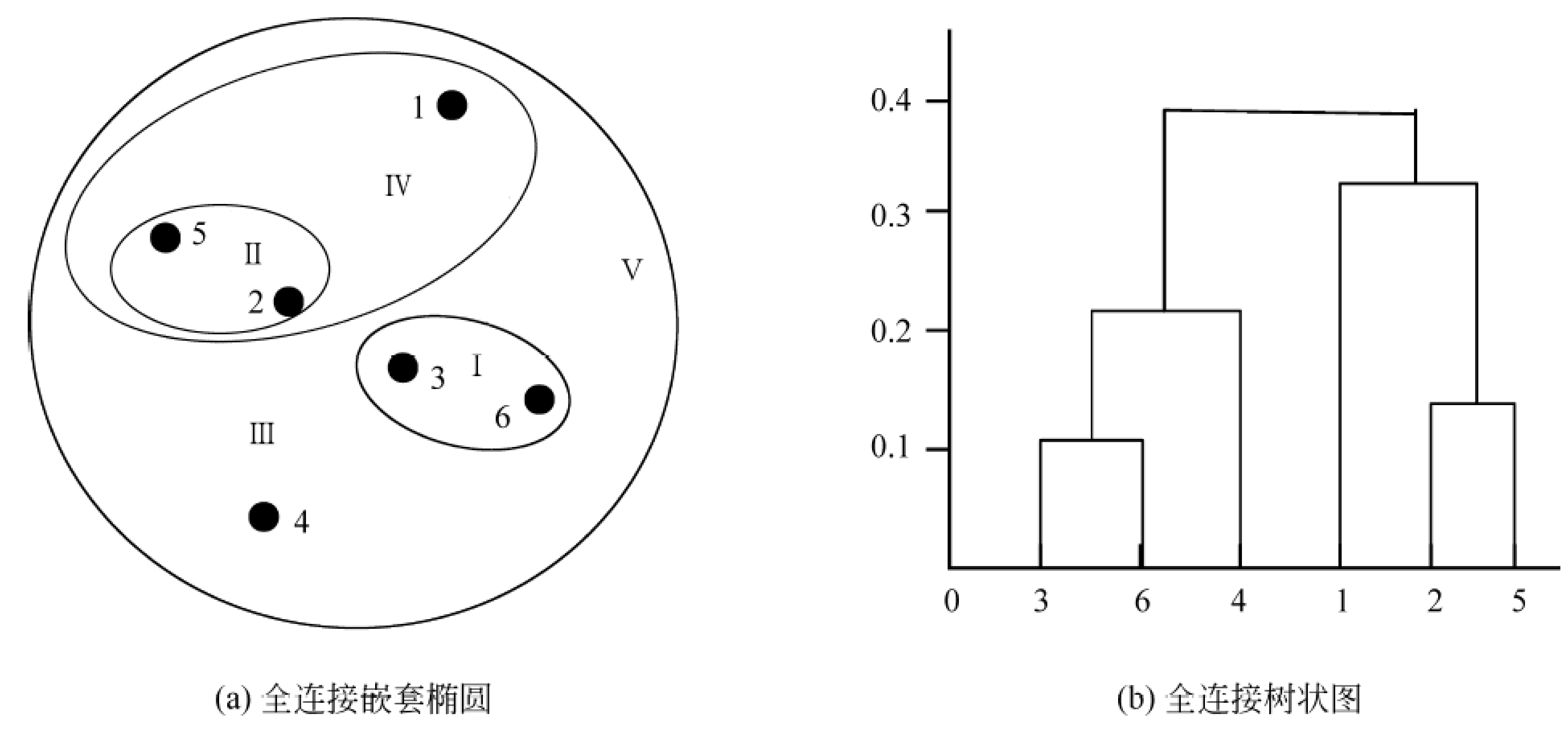
(a) 单连接嵌套椭圆　　(b) 单连接树状图

图 6-11　6 个数据点的单连接层次聚类

dist({3,6},{4})= max (dist(3,4),dist(6,4))
= max (0.15,0.22)
= 0.22

dist({3,6},{2,5})= max (dist(3,2),dist{6,2},dist(3,5),dist(6,5))
= max (0.15,0.25,0.28,0. +0.39)
= 0.39

dist({3,6},{1})= max (dist(3,1),dist(6,1))
= max (0.22,0.23)
= 0.23



(a) 全连接嵌套椭圆　　(b) 全连接树状图

图 6-12　6 个点的全连接层次聚类

(3) **平均连接(平均距离)**。对于层次聚类的平均连接度量方法,两个簇的距离定义为不同簇的所有点对距离的平均值。这是一种介于单连接和全连接之间的折中方法。

图 6-13 显示了将平均连接用于 6 个样本数据点集的结果。为了解释平均连接如何工作,计算某些簇之间的距离:

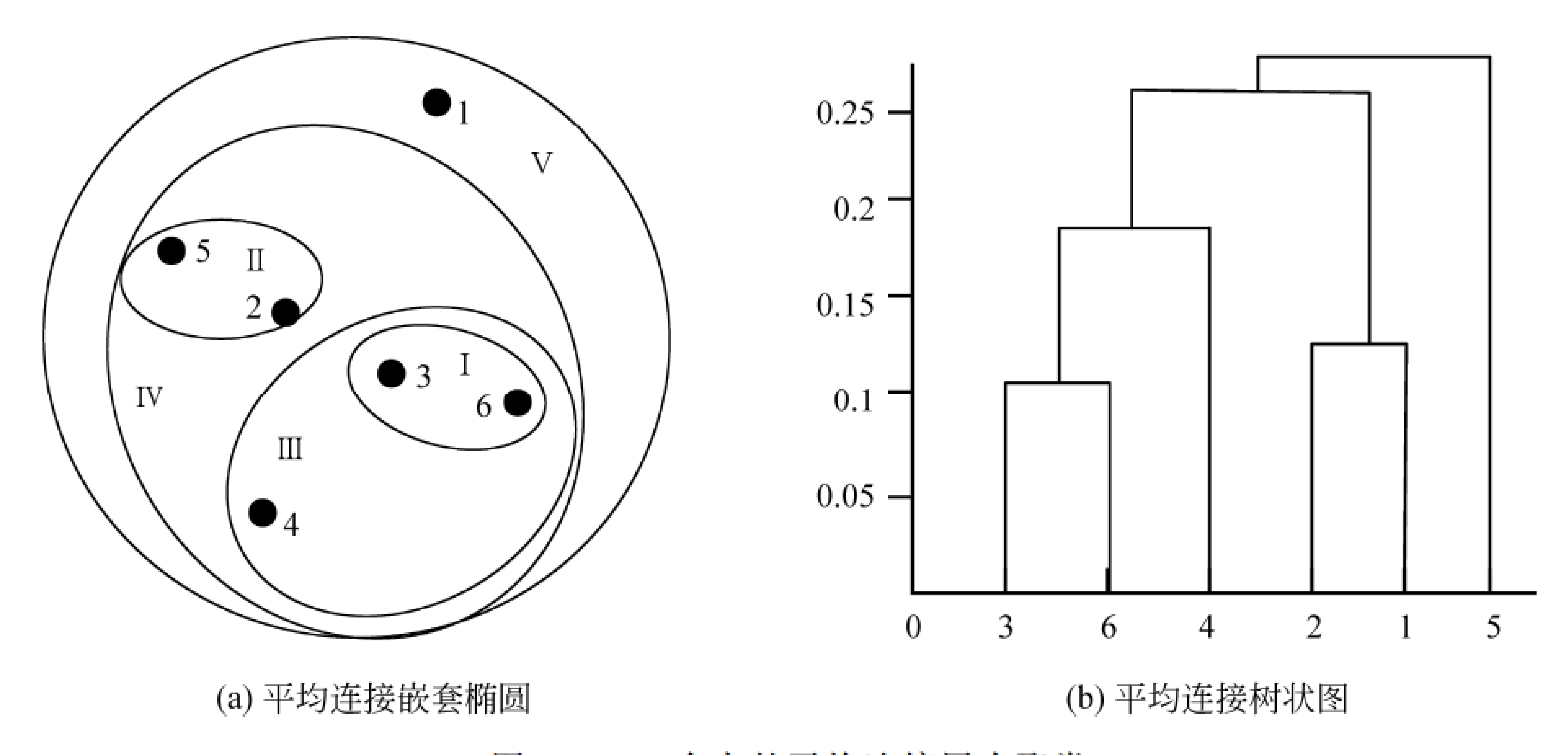
dist({3,6,4},{1}) = (0.22 + 0.37 + 0.23)/(3 × 1) = 0.28

dist({2,5},{1}) = (0.2357 + 0.3421)/(2 × 1) = 0.2889

dist({3,6,4},{2,5}) = (0.15 + 0.28 + 0.25 + 0.39 + 0.20 + 0.29)/(3 × 2)

= 0.26

因为 dist({3,6,4},{2,5})比 dist({3,6,4},{1})和 dist({2,5},{1})小,簇{3,6,4}和{2,5}在第 4 阶段合并。



(a) 平均连接嵌套椭圆　　(b) 平均连接树状图

图 6-13　6 个点的平均连接层次聚类

6.4 基于密度的方法

划分和层次方法旨在发现球状簇。它们很难发现任意形状的簇,如图 6-14 中 S 形和椭圆形簇。给定这种数据,它们很可能不正确地识别凸区域,其中噪声或离群点被包含在簇中。

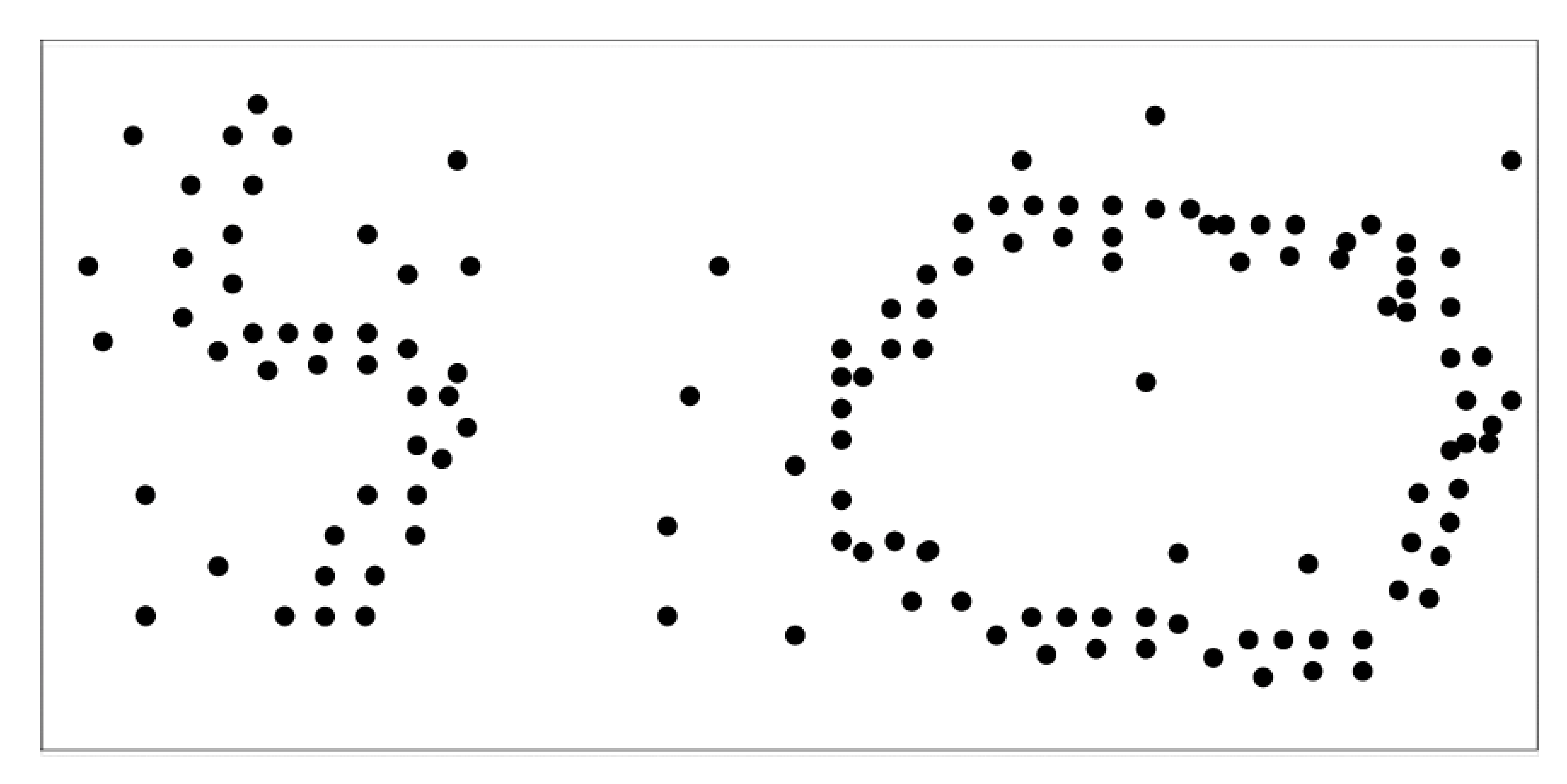


图 6-14　任意形状的簇

为了发现任意形状的簇,作为选择,可以把簇看作数据空间中被稀疏区域分开的稠密区

域。这是基于密度的聚类方法的主要策略,该方法可以发现非球状的簇。本节首先介绍密度的主要概念(6.4.1 节),介绍一种基于密度聚类的代表性方法——DBSCAN(6.4.2 节)。

6.4.1 传统的密度:基于中心的方法

尽管定义密度的方法没有定义相似度的方法多,但仍存在几种不同的方法。本节讨论 DBSCAN 使用的基于中心的方法。

在基于中心的方法中,数据集中特定点的密度通过对该点 Eps 半径之内的点计数(包括点本身)来估计,如图 6-15 所示。点 A 的 Eps 半径内点的个数为 7,包括 A 本身。

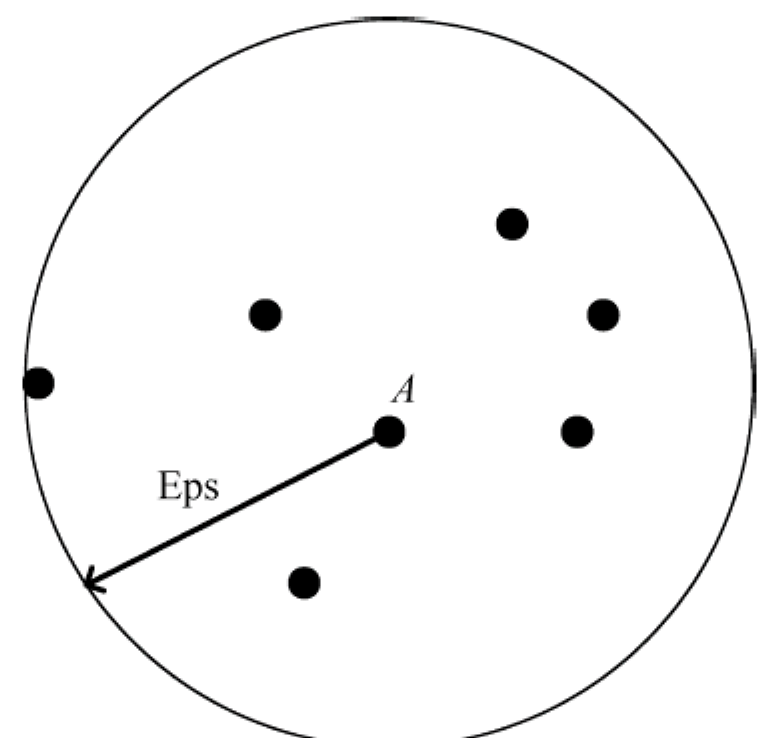


图 6-15 基于中心的密度

该方法实现简单,但是点的密度取决于指定的半径。例如,如果半径足够大,则所有点的密度都等于数据集中的点数 m。同理,如果半径太小,则所有点的密度都是 1。

根据基于中心的密度进行点分类,可以将点分类为稠密区域内部的点(核心点)、稠密区域边缘上的点(边界点)以及稀疏区域中的点(噪声点或背景点)。图 6-16 使用二维点集图示了核心点、边界点和噪声点的概念。

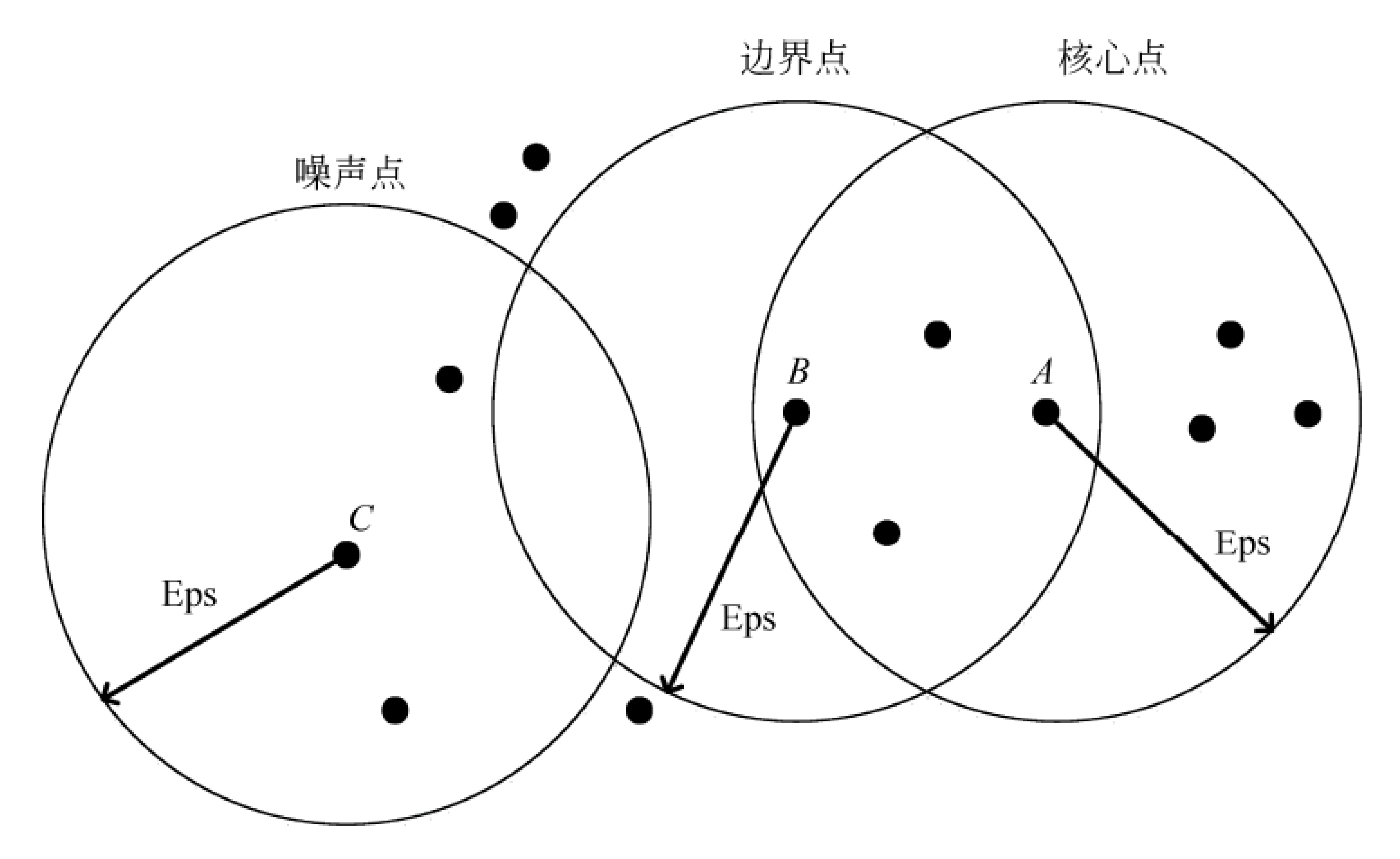


图 6-16 核心点、边界点和噪声点

- **核心点**(core point)。这些点在基于密度的簇内部。点的邻域由距离函数和用户指定的距离参数 Eps 决定。如果一个点的给定邻域内的点的个数超过给定的阈值 MinPts,其中 MinPts 也是一个用户指定的参数,则该点称为核心点。在图 6-16 中,点 A 是核心点。
- **边界点**(border point)。边界点不是核心点,但它落在某个核心点的邻域内。在图 6-16 中,点 B 是边界点。边界点可能落在多个核心点的邻域内。
- **噪声点**(noise point)。噪声点是既非核心点也非边界点的任何点。在图 6-16 中,

点 C 是噪声点。

给定核心点、边界点和噪声点的定义，下面解释 DBSCAN 所用到的一些基本术语。

- **对象的 ε-邻域**：给定对象在半径 ε 内的区域。
- **核心对象**(core object)。如果一个对象的 ε 邻域至少包含最小数目 MinPts 个对象，则称该对象为核心对象。
- **直接密度可达**(directly density-reachable)。给定一个对象集合 D，如果 p 是在 q 的 ε 邻域内，而 q 是一个核心对象，就说对象 p 从对象 q 出发是直接密度可达的。
- **密度可达的**(density-reachable)。如果存在一个对象链 $p_1,p_2,\cdots,p_n$，使得 $p_1=q,p_n=p$，并且对于 $p_i\in D(1\leqslant i\leqslant n)$，$p_{i+1}$ 是从 p_i 关于 ε 和 MinPts 直接密度可达的，则对象 p 是从对象 q 关于 ε 和 MinPts 密度可达的。
- **密度相连的**(density-connected)。如果对象集合 D 中存在一个对象 o，使得对象 p 和 q 是从 o 关于 ε 和 MinPts 密度可达的，那么对象 p 和 q 是关于 ε 和 MinPts 密度相连的。

例 6-5 密度可达和密度相连。

给定圆的半径为 ε，另 MinPts=3，考虑图 6-17。

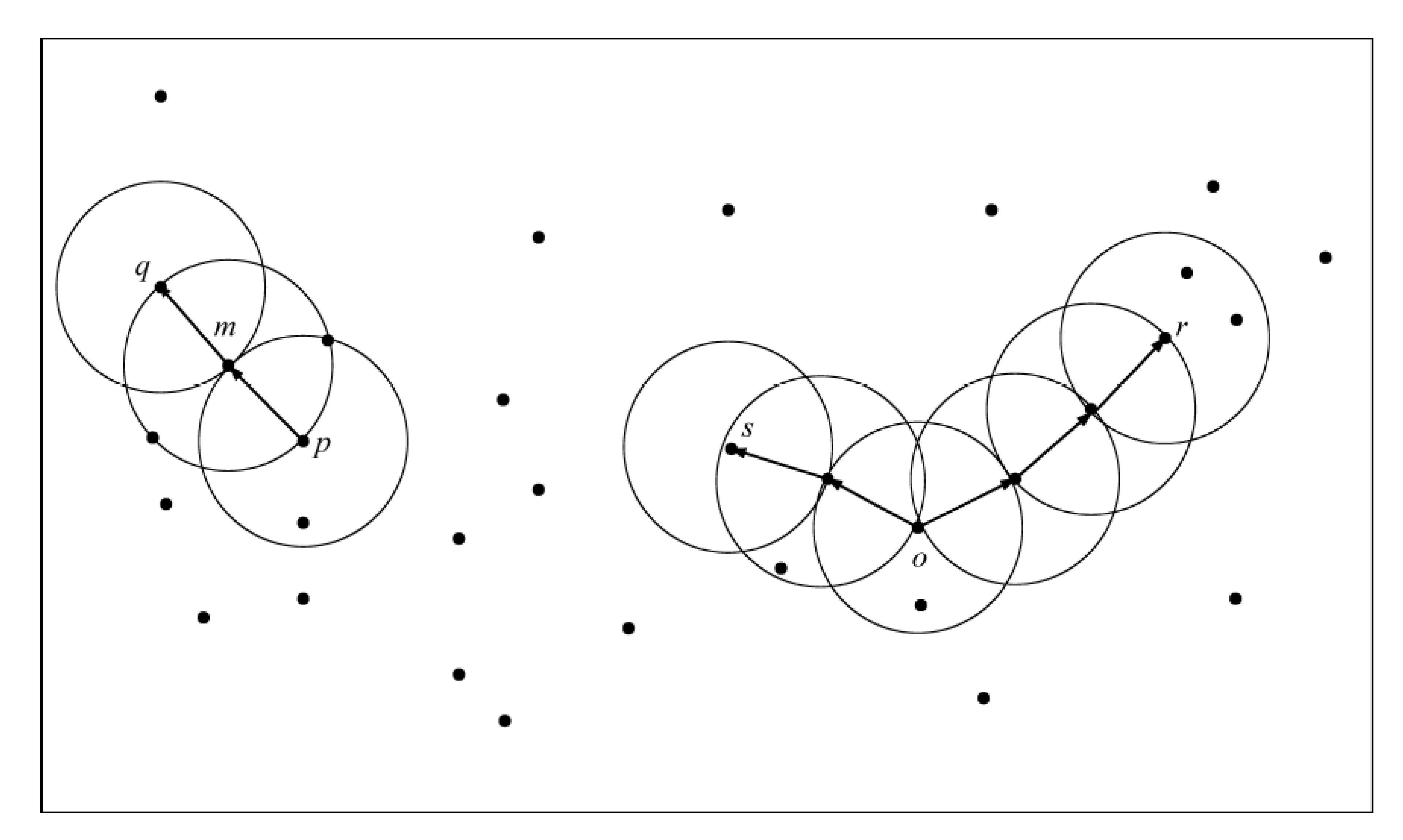


图 6-17 基于密度的聚类中的密度可达和密度相连

在被标记的点中，m、p、o 和 r 都是核心对象，因为它们的 ε 邻域内都至少包含 3 个对象。对象 q 是从 m 直接密度可达的。对象 m 是从 p 直接密度可达的，并且反之亦然。

对象 q 是从 p(间接)密度可达的，因为 q 是从 m 直接密度可达的，并且 m 是从 p 直接密度可达的。然而，p 并不是从 q 密度可达的，因为 q 不是核心对象。类似地，r 和 s 是从 o 密度可达的，而 o 是从 r 密度可达的。因此，o、r 和 s 都是密度相连的。

6.4.2 DBSCAN 算法

初始时，给定数据集 D 中的所有对象都被标记为 unvisited。DBSCAN 随机地选择一

个未访问的对象 p,标记 p 为 visited,并检查 p 的 ε 邻域是否至少包含 MinPts 个对象。如果不是,则 p 被标记为**噪声点**。否则,为 p 创建一个新的簇 C,并且把 p 的 ε 邻域中的所有对象都放到候选集合 N 中。DBSCAN 迭代地把 N 中不属于其他簇的对象添加到 C 中。在此过程中,对于 N 中标记为 unvisited 的对象 p',DBSCAN 把它标记为 visited,并且检查它的 ε 邻域。如果 p'的 ε 邻域至少有 MinPts 个对象,则 p'的 ε 邻域中的对象都被添加到 N 中。DBSCAN 继续添加对象到 C,直到 C 不能再扩展,即直到 N 为空。此时,簇 C 完全生成,于是被输出。

为了找出下一个簇,DBSCAN 从剩下的对象中随机地选择一个未访问的对象。聚类过程继续,直到所有对象都被访问。DBSCAN 算法的伪代码如算法 6-3 所示。

算法 6-3 DBSCAN,一种基于密度的聚类算法

输入:

D:包含 n 个对象的数据集

ε:半径参数

MinPts:邻域密度阈值

输出:基于密度的簇的集合

步骤:

```
(1) 标记所有对象为 unvisited;
(2) do
(3)     随机选择一个 unvisited 对象 p;
(4)     标记 p 为 visited;
(5)     if p 的ε邻域至少有 MinPts 个对象
(6)         创建一个新簇 C,并把 p 添加到 C;
(7)         令 N 为 p 的ε邻域中的对象的集合;
(8)         for N 中每个点 p'
(9)             if p'是 unvisited
(10)                标记 p'为 visited;
(11)                if p'的ε邻域至少有 MinPts 个点,把这些点添加到 N;
(12)            if p'还不是任何簇的成员,把 p'添加到 C;
(13)         end for
(14)         输出 C;
(15)    else 标记 p 为噪声;
(16) until 没有标记为 unvisited 的对象。
```

DBSCAN 的基本时间复杂度是 $O(n\times$找出 Eps 邻域中的点所需要的时间$)$,其中 n 是点的个数。在最坏情况下,时间复杂度是 $O(n^2)$。然而,在低维空间,有一些数据结构,如 kd 树,可以有效地检索特定点给定距离内的所有点,时间复杂度可以降低到 $O(n\log n)$。即使对于高维数据,DBSCAN 的空间也是 $O(n)$,因为对每个点,它只需要维持少量数据,即簇标号和每个点是核心点、边界点还是噪声点的标识。另外,如果用户定义的参数 ε 和 MinPts 设置得恰当,则该算法可以有效地发现任意形状的簇。

6.5 基于概率模型的聚类方法

迄今为止,在我们讨论的所有聚类分析方法中,每个数据对象只能被指派到多个簇中的一个。这种簇分配规则在某些应用中是必要的,如把客户分配给销售经理。然而,在其他应用中,这种僵硬的要求可能并非我们期望的,正如下面给出的例子所示。

例 6-6 聚类产品评论。

在一个网店中,顾客在在线购物的同时,可以对其产品发表评论。但并非每种产品都收到评论,而且某些产品可能有很多评论,而其他一些没有或很少。此外,一个评论可能涉及多种产品。这样,作为一名评论编辑,需要对这些评论进行聚类。

理想情况下,一个簇集中于一个主题,例如,一组产品、服务或高度相关的问题。对于评论编辑的任务而言,把评论互斥地指派到一个簇效果并不好。假设关于照相机和摄像机有一个簇,关于计算机有另一个簇。如果一个评论谈论到摄像机与计算机的兼容性,应该怎么指派?该评论与这两个簇都相关,而且并不互斥地属于任何一个簇。

此时需要使用一种聚类方法,它允许一个评论属于多个簇,如果该评论确是涉及多个主题的话。为了反映一个评论属于某个簇的强度,可以在评论到簇的指派上附加一个代表这种部分隶属关系的权重。

本节系统地研究允许一个对象属于多个簇的聚类主题。6.5.1 节讨论模糊聚类,6.5.2 节推广到基于概率模型的聚类,6.5.3 节介绍期望极大化算法。

6.5.1 模糊聚类

如果数据对象分布在明显分离的组中,则把对象明确划分成不相交的簇看来是一种理想的方法。然而,在大部分情况下,数据集中的对象不能划分成明显分离的簇,指派一个对象到一个特定的簇也具有一定的任意性。考虑一个靠近两个簇边界的对象,它离其中一个稍微近一点。在大多数这种情况下,下面的做法更合适:对每个对象和每个簇赋予一个权值,指明该对象属于该簇的程度。从数学上讲,w_{ij} 是对象 x_i 属于簇 C_j 的权值。

1. 模糊集合

1965 年,Lotfi Zadeh 引进模糊集合论(fuzzy set theory)和模糊逻辑(fuzzy logic)作为一种处理不精确和不确定性的方法。简要地说,模糊集合论允许对象以 0 和 1 之间的某个隶属度属于一个集合,而模糊逻辑允许一个陈述以 0 和 1 之间的确定度为真。传统的集合论和逻辑是对应的模糊集合论和模糊逻辑的特殊情况,它们限制集合的隶属度或确定度或者为 0,或者为 1。模糊概念已经用于许多不同的领域,包括控制系统、模式识别和数据分析(分类和聚类)。

考虑如下模糊逻辑的例子。陈述"天空有云"为真的程度可以定义为天空被云覆盖的百分比。例如,天空的 50%被云覆盖,则"天空多云"为真的程度是 0.5。如果有两个集合"多云天"和"非多云天",则可以类似地赋予每一天隶属于这两个集合的程度。这样,如果

一天 25%多云，则它在“多云天”集合中具有 0.25 的隶属度，而在“非多云天”集合中具有 0.75 的隶属度。

2. 模糊簇

给定一个对象集 $X=\{x_1,x_2,\cdots,x_n\}$，模糊集 S 是 X 的一个子集，它允许 X 中的每个对象都具有一个属于 S 的 0 到 1 之间的隶属度。形式地，一个模糊集 S 可以用一个函数 $F_s:X\to[0,1]$建模。

可以把模糊集概念用在聚类上。也就是说，给定对象的集合，一个簇就是对象的一个模糊集。这种簇称作**模糊簇**。因此，一个聚类包含多个模糊簇。

给定对象集 $o_1,o_2,\cdots,o_n$，k 个模糊簇 $C_1,C_2,\cdots,C_k$ 的模糊聚类可以用一个**隶属矩阵** $U=[w_{ij}](1\leqslant i\leqslant n,1\leqslant j\leqslant k)$表示。其中 w_{ij} 是 o_i 在模糊簇 C_j 的隶属度。隶属矩阵应该满足以下 3 个要求：

- 对于每个对象 o_i 和簇 C_j，$0\leqslant w_{ij}\leqslant 1$。这一要求强制模糊簇是模糊集。
- 对于每个对象 o_i，$\sum_{j=1}^{k} w_{ij}=1$。这一要求确保每个对象同等地参与聚类。
- 对于每个簇 C_j，$0<\sum_{i=1}^{n} w_{ij}<n$。这一要求确保对于每个簇，最少有一个对象，其隶属值非零。

3. 模糊 c 均值

尽管存在多种模糊聚类(事实上，许多数据分析算法都可以“模糊化”)，这里只考虑 k-means 的模糊版本，称作模糊 c 均值。模糊 c 均值算法有时称作 FCM，如算法 6-4 所示。

算法 6-4 模糊 c 均值

输入：

k：簇的数目

D：包含 n 个对象的数据集

$\boldsymbol{U}=[w_{ij}]$：隶属矩阵

输出：k 个簇的集合

步骤：

(1) 从 D 中任意选择 k 个对象作为初始簇中心；

(2) 用值在 0,1 之间的随机数初始化隶属矩阵 $\boldsymbol{U}$，对所有的 w_{ij} 赋值；

(3) repeat

(4) 　　根据每个对象的隶属度，将其分配到隶属度最高的簇；

(5) 　　更新每个簇的质心；

(6) 　　重新计算隶属矩阵；

(7) until 不再发生变化。

初始化之后，FCM 重复地计算每个簇的质心和隶属矩阵，直到划分不再改变。FCM 的结构类似于 k-means。k-means 在初始化之后，交替地更新质心和指派每个对象到最近

的簇。与 k-means 一样，FCM 可以解释为试图最小化误差的平方和(SSE)，尽管 FCM 是基于 SSE 的模糊版本。事实上，k-means 可以看作 FCM 的特例。FCM 的细节介绍如下。

(1) **计算 SSE**。误差的平方和(SSE)。将 SSE 的定义修改为

$$\mathrm{SSE}(C_1, C_2, \cdots, C_k) = \sum_{j=1}^{k} \sum_{i=1}^{n} w_{ij}^{p} \mathrm{dist}\,(x_i, c_j)^2 \tag{6-6}$$

其中 c_j 是第 j 个簇的质心，而 $p(p \geqslant 1)$控制隶属度的影响。p 的值越大，隶属度的影响越大。

(2) **初始化**。通常使用随机初始化。特殊地，权值随机地选取，同时限定与任何对象相关联的权值之和必须等于 1。与 k-means 一样，随机初始化是简单的，但是常常导致聚类结果代表 SSE 的局部最小。

(3) **计算质心**。式(6-7)给出的质心定义可以通过发现最小化公式(6-6)给定的模糊 SSE 的质心推导出来。对于簇 C_j，对应的质心 c_j 由下式定义：

$$c_j = \sum_{i=1}^{n} w_{ij}^{p} x_i \Big/ \sum_{i=1}^{n} w_{ij}^{p} \tag{6-7}$$

模糊质心的定义类似于传统的质心定义，不同之处在于所有点都要考虑，并且每个点对质心的贡献要根据它的隶属度加权。对于传统的明确集合，所有的 w_{ij} 或者为 0，或者为 1。

如果所选取的 p 值接近 1，则模糊 c 均值的行为很像传统的 k-means。另一方面，随着 p 增大，所有的簇质心都趋向于所有数据点的全局质心。换言之，随着 p 增大，划分变得越来越模糊。

(4) **更新隶属矩阵**。由于隶属矩阵有权值定义，因此这一步涉及更新与第 i 个点和第 j 个簇相关联的权值 w_{ij}。式(6-8)给出的权值更新公式可以通过限定权值之和为 1，最小化式(6-6)中的 SSE 导出。

$$w_{ij} = (1/\mathrm{dist}\,(x_i, c_j)^2)^{\frac{1}{p-1}} \Big/ \sum_{q=1}^{k} (1/\mathrm{dist}\,(x_i, c_q)^2)^{\frac{1}{p-1}} \tag{6-8}$$

直观地来看，权值 w_{ij} 指明点 x_i 在簇 C_j 中的隶属度。如果 x_i 靠近质心 c_j($\mathrm{dist}(x_i, c_j)$比较小)，则 w_{ij} 应当相对较高；而如果 x_i 远离质心 c_j($\mathrm{dist}(x_i, c_j)$比较大)，则 w_{ij} 相对较低。现在考虑式(6—8)中指数 $1/(p-1)$的影响。如果 $p>2$，则该指数降低赋予离点最近的簇的权值。事实上，随着 p 趋向于无穷大，该指数趋向于 0，而权值趋向于 $1/k$。另一方面，随着 p 趋向于 1，该指数加大赋予离点最近的簇的权值。随着 p 趋向于 1，关于最近簇的隶属权值趋向于 1，而关于其他簇的隶属权值趋向于 0。

6.5.2 基于概率模型的聚类

模糊簇提供了一种灵活性，允许一个对象属于多个簇。有没有一种说明聚类的一般框架，其中对象可以用概率的方法参与多个簇？本节介绍基于概率模型的聚类的一般概念来回答这一问题。

之所以在数据集上进行聚类分析，是因为假定数据集中的对象属于不同的固有类

别。这里，隐藏在数据中的固有类别是潜在的，因为不可能直接观测到它们，而必须使用观测数据来推测。例如，6.5节一开始提到的网店评论划分问题，评论集中的主题是潜在的，不能直接地了解到一个评论是属于哪一种主题。然而，可以从评论中推导出这些主题，因为每个评论都是关于一个或多个主题的。因此，聚类分析的目标是发现隐藏的类别。

从统计学讲，可以假定隐藏的类别是数据空间上的一个分布，可以使用概率密度函数（或分布函数）精确地表示。这种隐藏的类别称为**概率簇**（probabilistic cluster）。对于一个概率簇 C、它的密度函数 f 和数据空间的点 o，$f(o)$ 是 C 的一个实例在 o 上出现的相对似然。

假设要通过聚类分析找出 k 个聚类簇 $C_1, C_2, \cdots, C_k$。对于 n 个对象的数据集 D，可以把 D 看作这些簇的可能实例的一个有限样本。从概念上讲，可以假定 D 按如下方法形成。每个簇 $C_j(1 \leqslant j \leqslant k)$ 都与一个实例从该簇抽样的概率 w_j 相关联。通常假定 $w_1, w_2, \cdots, w_k$ 作为问题设置的一部分给定，并且 $\sum_{j=1}^{k} w_j = 1$，确保所有对象都被 k 个簇产生。

然后，运行如下两步，产生 D 的一个对象。这两个步骤总共执行 n 次，产生 D 的 n 个对象 $o_1, o_2, \cdots, o_n$。

(1) 按照概率 $w_1, w_2, \cdots, w_k$ 选择一个簇 C_j。

(2) 按照 C_j 的概率密度函数 f_j 选择一个 C_j 的实例。

该数据产生过程是混合模型的基本假定。**混合模型**将数据看作从不同的概率分布得到的观测值的集合。概率分布可以是任何分布，但是通常是多元正态的，因为这种类型的分布已被人们完全理解，容易从数学上进行处理，并且已经证明在许多情况下都能产生好的结果。

从概念上讲，混合模型对应于如下数据产生过程。给定几个分布（通常类型相同但参数不同），随机地选取一个分布并由它产生一个对象，重复该过程 n 次，其中 n 是对象的个数。

更形式化地，假定有 k 个分布和 n 个对象 $X=\{x_1, x_2, \cdots, x_n\}$。设第 j 个分布的参数为 θ_j，并设 Θ 是所有参数的集合，即 $\Theta=\{\theta_1, \theta_2, \cdots, \theta_k\}$。则 $P(x_i|\theta_j)$ 是第 i 个对象来自第 j 个分布的概率。选取第 j 个分布产生一个对象的概率由权值 $w_j(1 \leqslant j \leqslant k)$ 给定，其中权值（概率）受限于其和为1的约束，即 $\sum_{j=1}^{k} w_j = 1$。于是，对象 x 的概率由式(6-9)给出：

$$P(x \mid \Theta) = \sum_{j=1}^{k} w_j P_j(x \mid \theta_j) \tag{6-9}$$

如果对象以独立的方式产生，则整个对象集的概率是每个个体对象 x_i 的概率的乘积：

$$P(X \mid \Theta) = \prod_{i=1}^{n} P(x_i \mid \Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} w_j P_j(x_i \mid \theta_j) \tag{6-10}$$

对于混合模型，每个分布描述一个不同的组，即一个不同的簇。通过使用统计方法，可以由数据估计这些分布的参数，从而描述这些分布(簇)。也可以识别哪个对象属于哪个簇。然而，混合建模并不产生对象到簇的明确指派，而是给出具体对象属于特定簇的概率。

例 6-7 单变量的高斯混合模型。

用高斯分布给出混合模型的具体解释。一维高斯分布在点 x 的概率密度函数是

$$P(x_i \mid \Theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6-11}$$

该高斯分布的参数是 $\theta=(\mu,\sigma)$，其中 μ 是分布的均值，而 σ 是标准差。假定有两个高斯分布，具有共同的标准差 2，均值分别为 -4 和 4。还假定每个分布以等概率选取，即 $w_1 = w_2 = 0.5$。于是式(6-9)变成

$$P(x \mid \Theta) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x+4)^2}{8}} + \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-4)^2}{8}} \tag{6-12}$$

图 6-18 显示该混合模型的概率密度函数，使用单变量高斯混合模型的基于概率模型的聚类分析任务是推断 Θ，使得式(6-12)最大化。

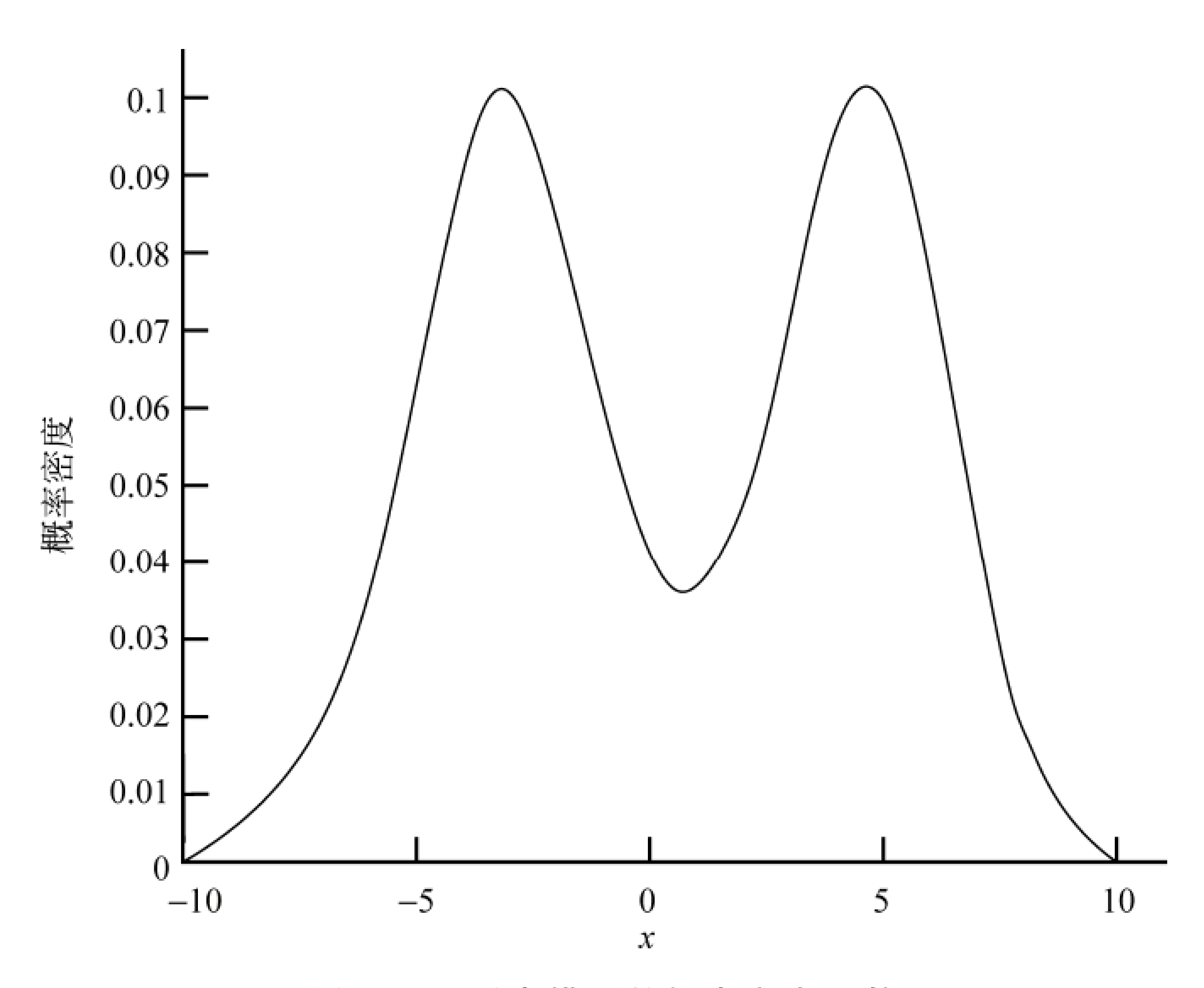


图 6-18 混合模型的概率密度函数

6.5.3 期望最大化算法

如何计算模糊聚类和基于概率模型的聚类？本节介绍一种原理性方法。

k-means 聚类可以看作是模糊聚类的一种特例，k-means 算法迭代执行直到不能再改进聚类。每次迭代包括两个步骤：

(1) **期望步(E-步)**。给定当前的簇中心，每个对象都被指派到簇中心离该对象最近的簇。这里，期望每个对象都属于最近的簇。

（2）**最大化步（M-步）**。给定簇指派，对于每个簇，算法调整其中心，使得指派到该簇的对象到该新中心的距离之和最小化。也就是说，将指派到一个簇的对象的相似度最大化。

可以推广这两步过程来处理模糊聚类和基于概率模型的聚类。一般而言，**期望-最大化**（Expectation-Maximization，EM）算法是一种框架，它逼近统计模型参数的最大似然或最大后验估计。在模糊或基于概率模型的聚类的情况下，EM 算法从初始参数集出发，并且迭代直到不能改善聚类，即直到聚类收敛或改变充分小（小于一个预先设定的阈值）。每次迭代也由两步组成：

（1）**期望步**根据当前的模糊聚类或概率簇的参数把对象指派到簇中。

（2）**最大化步**发现新的聚类或参数，最大化模糊聚类的 SSE 或基于概率模型的聚类的期望似然。

例 6-8　EM 算法的简单例子。

这个例子解释 EM 算法用于图 6-18 的数据时如何执行。为了使这个例子尽可能简单，假定已知两个分布的标准差都是 2.0，并且点以相等的概率由两个分布产生。把左边和右边的分布分别称作分布 1 和分布 2。

从对 μ_1 和 μ_2 的初始猜测开始 EM 算法，例如，取 $\mu_1=-2$，$\mu_2=3$。这样，对于两个分布，初始参数 $\theta=(\mu,\sigma)$分别是 $\theta_1=(-2,2)$和 $\theta_2=(3,2)$。整个混合模型的参数集是 $\Theta=\{\theta_1,\theta_2\}$。对于 *EM* 的期望步，要计算某个点取自一个特定分布的概率，即，要计算 P(分布 1$|x_i,\Theta$)和 P(分布 2$|x_i,\Theta$)。这些值可以用下式表示，它是贝叶斯规则的直接应用。

$$P(\text{分布 } j \mid x_i,\theta)=\frac{0.5P(x_i \mid \theta_j)}{0.5P(x_i \mid \theta_1)+0.5P(x_i \mid \theta_2)} \tag{6-13}$$

其中，0.5 是每个分布的概率（权），而 j 是 1 或 2。

例如，假定其中一个点是 0。计算 $P(0|\theta_1)=0.12$，$P(0|\theta_2)=0.06$（实际计算的是概率密度）。使用这些值和式（6-13），发现 P(分布 1$|0,\Theta$)$=0.12/(0.12+0.06)=0.66$，P(分布 2$|0,\Theta$)$=0.06/(0.12+0.06)=0.33$。根据对参数值的当前假设，这意味着点 0 属于分布 1 的可能性是属于分布 2 的可能性的两倍。

计算了 20 000 个点的簇隶属概率之后，在 EM 算法的最大化步计算 μ_1 和 μ_2 的新的估计（使用式（6-14）和式（6-15））。注意，分布的均值的新的估计是点的加权平均，其中权值是点属于该分布的概率，即值 P(分布 $j|x_i$)。

$$\mu_1=\sum_{i=1}^{20\,000} x_i \frac{P(\text{分布 } 1 \mid x_i,\Theta)}{\sum_{i=1}^{20\,000} P(\text{分布 } 1 \mid x_i,\Theta)} \tag{6-14}$$

$$\mu_2=\sum_{i=1}^{20\,000} x_i \frac{P(\text{分布 } 2 \mid x_i,\Theta)}{\sum_{i=1}^{20\,000} P(\text{分布 } 2 \mid x_i,\Theta)} \tag{6-15}$$

重复这两步，直到 μ_1 和 μ_2 的估计不再改变或变化很小。表 6-2 显示 EM 算法用于 20 000 个点的集合的前几次迭代。对于该数据，我们知道哪个分布产生哪些点，因此也可以由每个分布计算均值。这些均值是 $\mu_1=-3.98$ 和 $\mu_2=4.03$。

表 6-2 简单例子 EM 算法的前几次迭代

迭代	μ_1	μ_2	迭代	μ_1	μ_2
0	−2.00	3.00	3	−3.97	4.04
1	−3.74	4.10	4	−3.98	4.03
2	−3.94	4.07	5	−3.98	4.03

在许多应用中，基于概率模型的聚类已经表现出很好的效果，因为它比划分方法和模糊聚类方法更通用。它的一个突出优点是使用合适的统计模型以捕获潜在的簇。EM 算法因其简洁性，已经广泛用来处理数据挖掘和统计学的许多学习问题。注意，一般而言，EM 算法可能不会收敛到最优解，而是可能收敛于局部最大。研究者已经考察了许多避免收敛于局部极大的启发式方法。例如，可以使用不同的随机初始值多次运行 EM 过程。此外，如果分布很多或数据集只包含很少观测数据点，则 EM 算法的计算开销可能很大。

6.6 聚类评估

到目前为止，我们已经学习了什么是聚类，并且已经认识了一些常见的聚类方法。你可能会问："当我们在数据集上使用一种聚类方法时，如何评估聚类的结果是否好？"一般而言，**聚类评估**用于对在数据集上进行聚类的可行性和被聚类方法产生的结果的质量进行评估。聚类评估主要包括以下任务：

- **估计聚类趋势**。在这项任务中，对于给定的数据集，评估该数据集是否存在非随机结构。如果盲目地在数据集上使用聚类方法返回一些簇，所挖掘的簇可能是误导。数据集上的聚类分析仅当数据中存在非随机结构时才是有意义的。
- **确定数据集中的簇数**。一些诸如 k-means 这样的算法需要数据集的簇数作为参数。此外，簇数可以看作数据集的有趣并且重要的概括统计量。因此，在使用聚类算法导出详细的簇之前，估计簇数是可取的。
- **测定聚类质量**。在数据集上使用聚类方法之后，想要评估结果簇的质量，许多度量都可以使用。有些方法测定簇对数据的拟合程度，而其他方法测定簇与基准匹配的程度，如果这种基准存在的话。还有一些方法对聚类打分，因此可以比较相同数据集上的两组聚类结果。

6.6.1 聚类趋势的估计

聚类趋势评估确定给定的数据集是否具有可以导致有意义的聚类的非随机结构。考虑一个没有任何非随机结构的数据集，如图 6-19 中数据空间中均匀分布的点。尽管聚类算法可以为该数据集返回簇，但是这些簇是随机的，没有任何意义。

为了处理这样的问题，可以使用多种算法来评估结果簇的质量。如果簇都很差，则可能表明数据中确实没有簇。

换一种方式，可以关注聚类趋势度量，即试图评估数据集中是否包含簇，而不进行聚

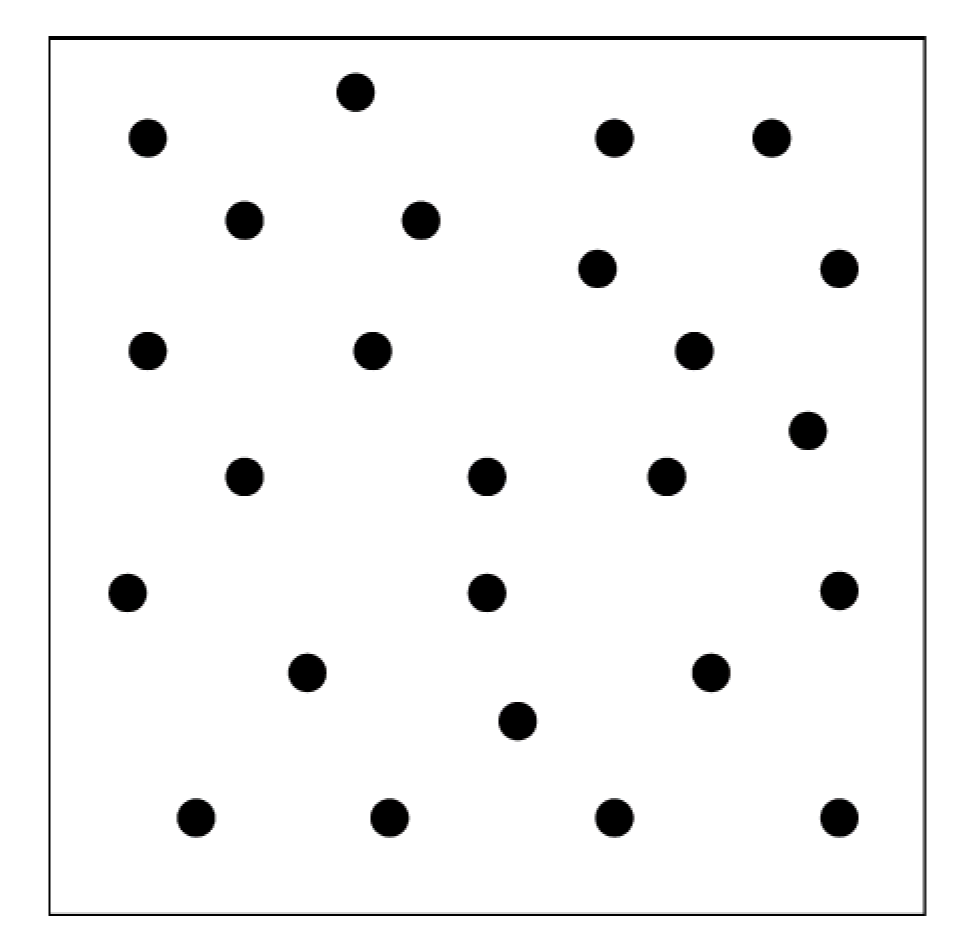


图 6-19 一个在数据空间均匀分布的数据集

类。最常用的方法(特别是对于欧几里得空间数据)是使用统计检验来检验空间随机性。然而,选择正确的模型,估计参数,评估数据是非随机的假设的统计数据,这一切可能非常具有挑战性。尽管如此,人们已经开发了许多方法,其中大部分都是针对低维欧几里得空间中的点。为了解释这一思想,下面考察一种简单但有效的统计量——霍普金斯统计量。

霍普金斯统计量(Hopkins Statistic)是一种空间统计量,检验空间分布的变量的空间随机性。给定数据集 D,它可以看作随机变量 o 的一个样本,我们想要确定 o 在多大程度上不同于数据空间中的均匀分布。可以按以下步骤计算霍普金斯统计量:

(1) 从 D 的空间中随机产生 n 个点 $p_1, p_2, \cdots, p_n$。也就是说,D 的空间中的每个点都以相同的概率包含在这个样本中。对于每个点 $p_i(1 \leqslant i \leqslant n)$,找出 p_i 在 D 中的最近邻,并令 x_i 为 p_i 与它在 D 中的最近邻之间的距离,即

$$x_i = \min_{v \in D}\{\mathrm{dist}(p_i, v)\} \tag{6-16}$$

(2) 均匀地从 D 中抽取 n 个点 $q_1, q_2, \cdots, q_n$。对于每个点 $q_i(1 \leqslant i \leqslant n)$,找出 q_i 在 $D-\{q_i\}$中的最近邻,并令 y_i 为 q_i 与它在 $D-\{q_i\}$中的最近邻之间的距离,即

$$y_i = \min_{v \in D, v \neq q_i}\{\mathrm{dist}(q_i, v)\} \tag{6-17}$$

(3) 计算霍普金斯统计量 H:

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i} \tag{6-18}$$

如果 D 是均匀分布的,则 $\sum_{i=1}^{n} x_i$ 和 $\sum_{i=1}^{n} y_i$ 将会很接近,因为 H 大约为 0.5。H 值接近 0 或 1 分别表明数据是高度聚类的和数据在数据空间是有规律分布的。为了举例说明,对于 $p=20$ 和 100 的不同实验,计算了图 6-19 中数据的霍普金斯统计量。H 的平均值为 0.56,标准差为 0.03。

原假设是同质假设,即 D 是均匀分布的,因而不包含有意义的簇。非均匀假设(即 D

不是均匀分布的，因而包含簇）是备择假设。可以迭代地进行霍普金斯统计量检验，使用0.5作为拒绝备择假设阈值，即如果 $H>0.5$，则 D 不大可能具有统计显著的簇。

6.6.2 聚类簇数的确定

确定数据集中“正确的”簇数是重要的，不仅因为像 k-means 这个的聚类算法需要这个参数，而且因为合适的簇数可以控制适当的聚类分析粒度。这可以看作在聚类分析的**可压缩性**与**准确性**之间寻找好的平衡点。考虑两种极端情况，如果把整个数据集看作一个簇会怎么样？这将最大化数据的压缩，但是这种聚类分析没有任何价值。另一方面，把数据集的每个对象看作一个簇将产生最细的聚类（即最准确的解，由于对象到其对应的簇中心的距离都为0）。在像 k-means 这样的算法中，这甚至实现了开销最小。然而，每个簇一个对象并不提供任何数据概括。

确定簇数并非易事，因为“正确的”簇数常常是含糊不清的。通常，找出正确的簇数依赖于数据集分布的形状和尺度，也依赖于用户要求的聚类分辨率。有许多估计簇数的可能方法。这里，简略介绍几种简单但流行和有效的方法。

一种简单的**经验方法**是，对于 n 个点的数据集，设置簇数 p 大约为 $\sqrt{\frac{n}{2}}$。在期望情况下，每个簇大约有 $\sqrt{2n}$ 个点。

拐点法（elbow method）基于如下观察：增加簇数有助于降低每个簇的簇内方差之和。这是因为有更多的簇可以捕获更细的数据对象簇，簇中对象之间更为相似。然而，如果形成太多的簇，则降低簇内方差和的边缘效应可能下降，因为把一个凝聚的簇分裂成两个簇只能使簇内方差和的稍微降低。因此，一种选择正确的簇数启发式方法是使用簇内方差和关于簇数的曲线的拐点。

例 6-9 簇的个数。

假设一个数据集有10个自然簇。图6-20显示了该数据集的 k-means 聚类发现的簇数的 SSE 曲线，而图6-21显示了相同数据的簇数的平均轮廓系数曲线。当簇数等于10时，SSE 有一个明显的拐点，而轮廓系数有一个明显的尖峰。

当然，这种方法并不总是有效的，有时簇可能盘绕或交叠得比较厉害。此外，数据中也可能包含嵌套的簇。

更高级的方法是使用信息准则或信息论的方法确定簇数。更多资料请参阅本章的参考文献（6.10节）。

数据集中“正确的”簇数还可以通过**交叉验证**确定。交叉验证是一种常用于分类的技术。首先，把给定的数据集 D 划分成 m 个部分。然后，使用 $m-1$ 个部分建立一个聚类模型，并使用剩下的一部分检验聚类的质量。例如，对于检验集中的每个点，可以找出最近的质心。因此，可以使用检验集中的所有点与它们的最近形心之间的距离的平方和来度量聚类模型拟合检验集的程度。对于任意整数 $k>0$，依次使用每一部分作为检验集，重复以上过程 m 次，导出 k 个簇的聚类。取质量度量的平均值作为总体质量度量。然后，对不同的 k 值，比较总体质量度量，并选取最佳拟合数据的簇数。

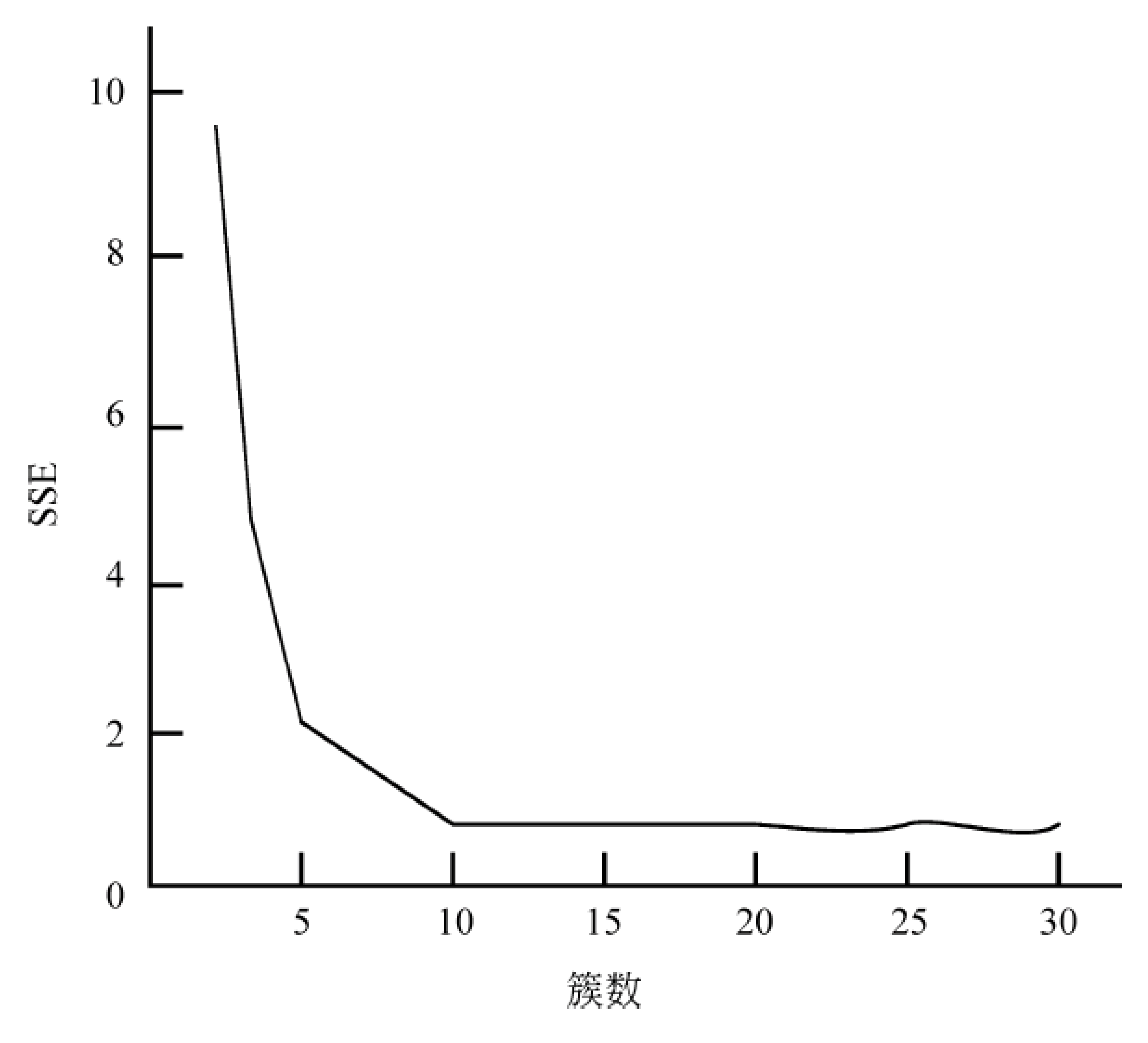


图 6-20　10 个自然簇数据集的 SSE 曲线

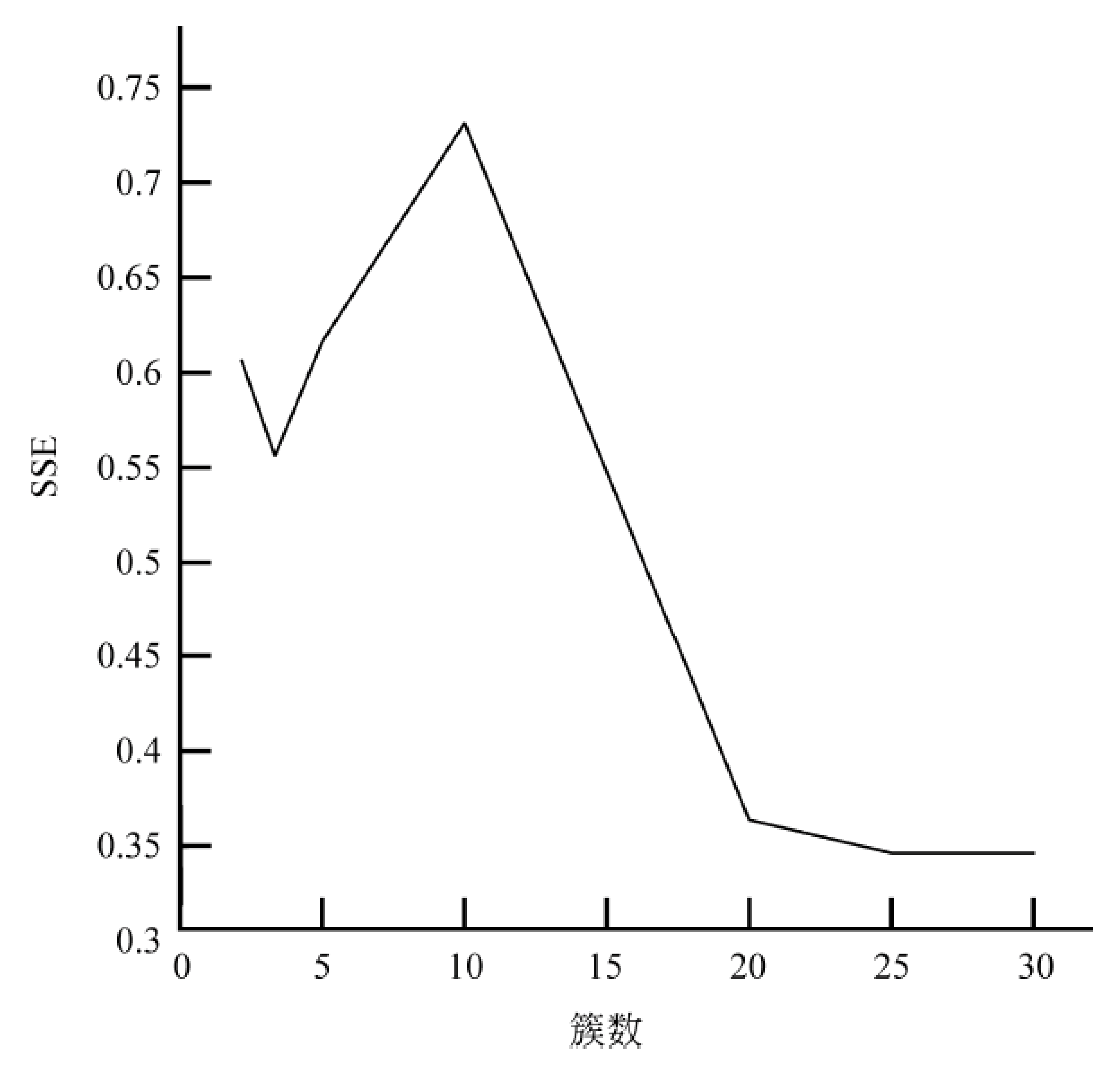


图 6-21　10 个自然簇数据集的平均轮廓系数曲线

6.6.3　聚类质量的测定

当得到有关于数据的外部信息(通常是从外部导出的数据对象的类标号形式)时,在这种情况下,惯常的做法是度量簇标号与类标号的对应程度。但是,这样做的目的是什么?归根结底,如果有了类标号,进行聚类分析的目的何在?这种分析的动机是比较聚类技术与“基本事实”,或评估人工分类过程可以在多大程度上被聚类分析自动地实现。

一般而言，根据是否有基准（基准是一种理想的聚类，通常由专家构建）可用，可以有两类不同的聚类质量测定方法。如果有可用的基准，则可以使用**外在方法**（extrinsic method），外在方法比较聚类结果和基准。如果没有基准可用，则可以使用**内在方法**（intrinsic method），通过考虑簇的分离情况评估聚类的好坏。基准可以看作一种“簇标号”形式的监督。因此，外在方法又称监督方法，而内在方法是无监督方法。

1. 外在方法

有许多度量（如熵、纯度、精度、召回率和 F 度量）普遍用来评估分类模型的性能。对于分类，度量预测的类标号与实际类标号的对应程度，但是对于上面提到的度量，通过使用簇标号而不是预测的类标号，不需要做重大改变。下面简略地介绍这些度量的定义。

- **熵**。每个簇由单个类的对象组成的程度。对于每个簇，首先计算数据的类分布，即对于簇 i，计算簇 i 的成员属于类 j 的概率 $p_{ij}=m_{ij}/m_i$，其中 m_i 是簇 i 中对象的个数，而 m_{ij} 是簇 i 中类 j 的对象个数。使用类分布，用标准公式 $e_i=-\sum_{j=1}^{L}p_{ij}\log_s p_{ij}$ 计算每个簇 i 的熵，其中 L 是类的个数。簇集合的总熵用每个簇的熵的加权和计算，即 $e=\sum_{i=1}^{K}\frac{m_i}{m}e_i$，其中 K 是簇的个数，而 m 是数据点的总数。
- **纯度**。簇包含单个类的对象的另一种度量程度。使用前面的术语，簇 i 的纯度是 $p_i=\max_j p_{ij}$，而聚类的总纯度是 $\text{purity}=\sum_{i=1}^{K}\frac{m_i}{m}p_i$。
- **精度**。簇中一个特定类的对象所占的比例。簇 i 关于类 j 的精度是 $\text{precision}(i,j)=p_{ij}$。
- **召回率**。簇包含一个特定类的所有对象的程度。簇 i 关于类 j 的召回率是 $\text{recall}(i,j)=m_{ij}/m_j$，其中 m_j 是类 j 的对象个数。
- ***F* 度量**。精度和召回率的组合，度量在多大程度上簇只包含一个特定类的对象和包含该类的所有对象。簇 i 关于类 j 的 F 度量是

$$F(i,j)=(2\times\text{precision}(i,j)\times\text{recall}(i,j))/(\text{precision}(i,j)+\text{recall}(i,j))$$

例 6-10 监督评估度量。

下面提供一个例子解释这些度量。具体地说，以余弦相似性度量使用 k-means，对取自《洛杉矶时报》的 3204 篇报道文章进行聚类。这些文章取自 6 个不同的类：娱乐、财经、国外、国内、都市和体育。表 6-3 显示了 k-means 聚类发现 6 个簇的结果。第一列指示簇，接下来的 6 列形成混淆矩阵，则这些列指出每个类的文档在这些簇中如何分布。最后两列分别是熵和纯度。

表 6-3 《洛杉矶时报》文档数据集 *k*-means 聚类结果

簇	娱乐	财经	国外	国内	都市	体育	熵	纯度
1	3	5	40	96	506	27	1.2270	0.7474
2	4	7	280	39	29	2	1.1472	0.7756

续表

簇	娱乐	财经	国外	国内	都市	体育	熵	纯度
3	1	1	1	4	7	671	0.1813	0.9796
4	10	162	3	73	119	2	1.7487	0.4390
5	331	22	5	13	70	23	1.3976	0.7134
6	5	358	12	48	212	13	1.5523	0.5525
合计	354	555	341	273	943	738	1.1450	0.7203

理想情况下，每个簇仅包含来自一个类的文档。事实上，每个簇包含来自多个类的文档。尽管如果，许多簇包含的文档主要来自一个类。具体地说，簇 3 包含的文档大部分来自体育版，纯度和熵都异常好。其他簇的纯度和熵没有这么好，但是如果数据被划分到更多的簇，则簇的纯度和熵可能大幅度提高。

可以对每个簇计算精度、召回率和 F 度量。为了给出一个具体的例子，考虑表 6-3 的簇 1 和都市类。精度是 506/677＝0.75，召回率是 506/943＝0.26，F 值是 0.39。相比之下，簇 3 和体育的 F 值是 0.94。

2. 内在方法

当没有数据集的基准可用时，必须使用内在方法来评估聚类的质量。一般而言，内在方法通过考察簇的分离情况和簇的紧凑情况来评估聚类。许多内在方法都利用数据集的对象之间的相似性度量。

轮廓系数(silhouette coefficient)就是这种度量。对于 n 个对象的数据集 D，假设 D 被划分成 k 个簇 $C_1,C_2,\cdots,C_k$。对于每个对象 $o\in D$，计算 o 与其所属的簇的其他对象之间的平均距离 $a(o)$。类似地，$b(o)$是 o 到不属于 o 的所有簇的最小平均距离。假设 $o\in C_i$ $(1\leqslant i\leqslant k)$，则

$$a(o)=\frac{\sum\limits_{o'\in C_i,o\neq o'}\text{dist}(o,o')}{|C_i|-1} \tag{6-19}$$

而

$$b(o)=\min_{C_j:1\leqslant j\leqslant k,j\neq i}\left\{\frac{\sum\limits_{o'\in C_j}\text{dist}(o,o')}{|C_j|}\right\} \tag{6-20}$$

对象 o 的轮廓系数定义为

$$s(o)=\frac{b(o)-a(o)}{\max\{a(o),b(o)\}} \tag{6-21}$$

轮廓系数的值在－1 和 1 之间。$a(o)$的值反映 o 所属的簇的紧凑性。该值越小，簇越紧凑。$b(o)$的值捕获 o 与其他簇的分离程度。$b(o)$的值越大，o 与其他簇越分离。因此，当 o 的轮廓系数值接近 1 时，包含 o 的簇是紧凑的，并且 o 远离其他簇，这是一种可取的情况。然而，当轮廓系数的值为负时(即 $b(o)<a(o)$)，这意味着在期望情况下，o 距离

其他簇的对象比距离与自己同在簇的对象更近。在许多情况下，这是很糟糕的，应该避免。

为了度量聚类中的簇的拟合性，可以计算簇中所有对象的轮廓系数的平均值。为了度量聚类的质量，可以使用数据集中所有对象的轮廓系数的平均值。通过启发式地导出数据集的簇数取代簇内方差之和，轮廓系数和其他内在度量也可以用在拐点法中。

6.7 基于 Python 平台的案例分析

联合国统计委员会每年都要统计全世界各个国家的人口出生和死亡情况，并给出人口统计年鉴用以分析人口趋势以及人口变动反映出的地区发展情况。研究发现，不同出生率和死亡率的国家或地区往往是呈现地域分布的，人口出生率和死亡率与该地域的发展有着密切关系。从出生率的分布可以看出一个国家的经济状况和文化状况，从死亡率的分布则可以折射出一个国家的营养卫生条件或其他社会性原因。

接下来，应用 Python 平台，以各国家的人口变动情况为背景，对不同国家的出生率和死亡率情况进行 *k*-means 聚类，希望能挖掘出其背后隐藏的信息。

6.7.1 数据准备

现在，假如联合国统计委员会调查了 215 个国家的人口出生和死亡情况，数据集命名为 birth-death-rates。每个国家分别作为一个样本数据占据数据集的一行，总共有 215 行，其中每条数据主要包含了以下 3 种特征：

- 国家的名称。
- 年平均出生率。
- 年平均死亡率。

表 6-4 给出了前 9 个国家的年平均出生率和死亡率。

表 6-4 各国出生率和死亡率数据集

Country	birth-rates	death-rates
Afghanistan	39.3	14.59
Albania	12.38	6.25
Algeria	24.4	4.3
American Samoa	22.9	4.6
Andorra	9.26	6.52
Angola	39.36	12.06
Anguilla	12.9	4.41
Antigua and Barbuda	16.19	5.72
Argentina	17.34	7.36

在将上述特征数据输入到聚类器之前，必须将待处理数据的格式改变为聚类器可以接受的格式，删除不需要的特征。首先，只取数据集中的 birth-rates 和 death-rates 两个特征，而 Country 特征则标识为每条数据的标号，并不参与聚类过程。然后，用 Python 命令加载数据集，如图 6-22 所示。

```
#导入数据集
df=pd.read_csv('birth-Death-rates.csv')
data=df.values[:, 1:].tolist()#过滤前两列
for i in range(len(data)):
      for j in range(len(data[0])):
            if type(data[i][j]) is types.LongType:
                  data[i][j]=float(data[i][j])
```

图 6-22　导入数据集命令

6.7.2　聚类分析结果探讨

现在，有一个符合标准的出生率、死亡率情况列表存储在 birth-death-rates. csv 中。接下来，就开始通过 *k*-means 聚类系统对这些国家进行聚类分析。

首先，需要制作原始数据的散点图，如图 6-23 所示。

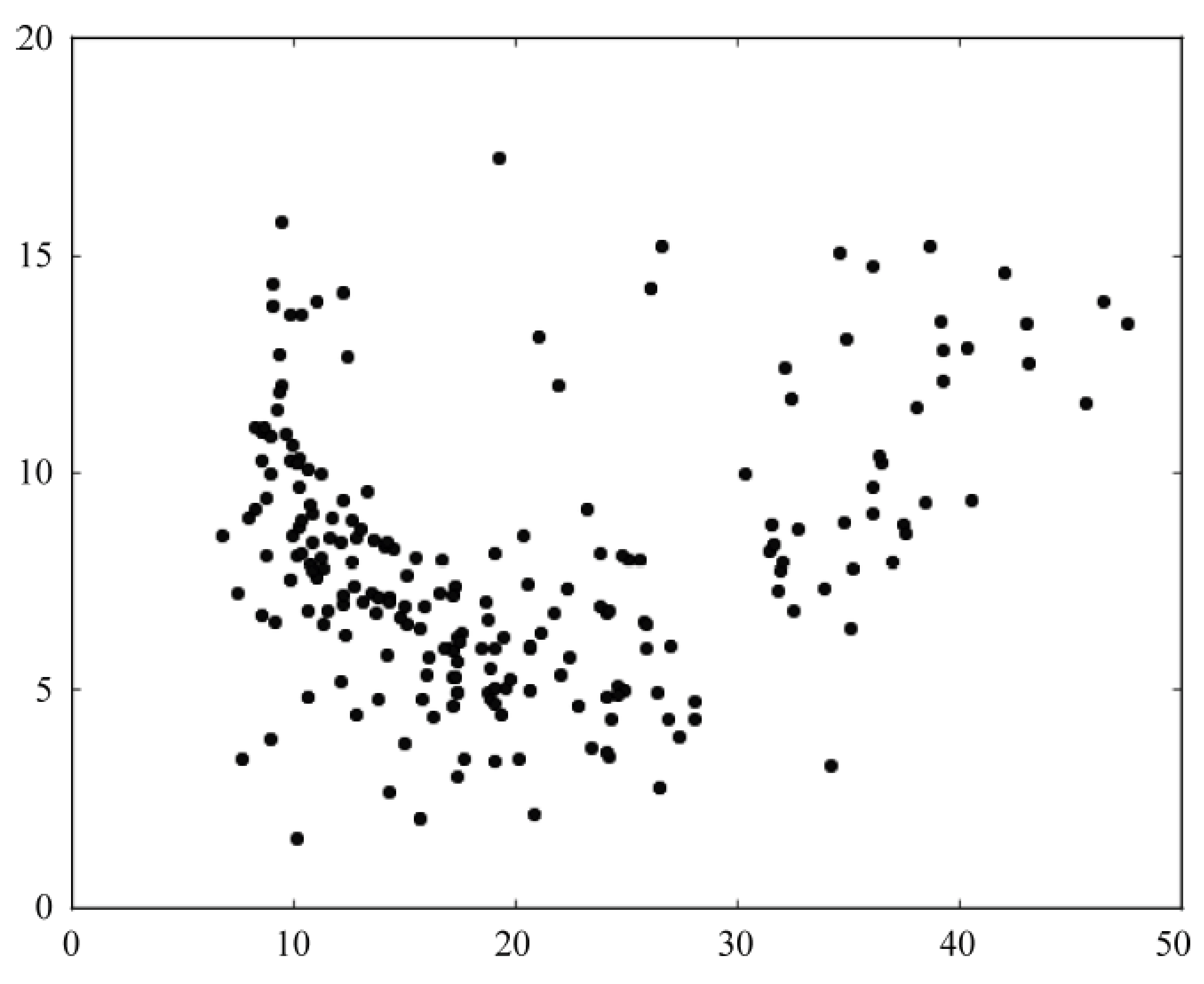


图 6-23　不同国家年平均出生率和死亡率分布情况

散点图的横坐标和纵坐标分别表示特征值 birth-rates 和 death-rates，从图中也可看出数据的大致分布情况。

在系统初始界面中，需要确定需要的特征个数以及对象个数。在这里，只考虑 birth-rates 和 death-rates 两个特征，选取 215 个国家作为聚类对象，如图 6-24 所示。

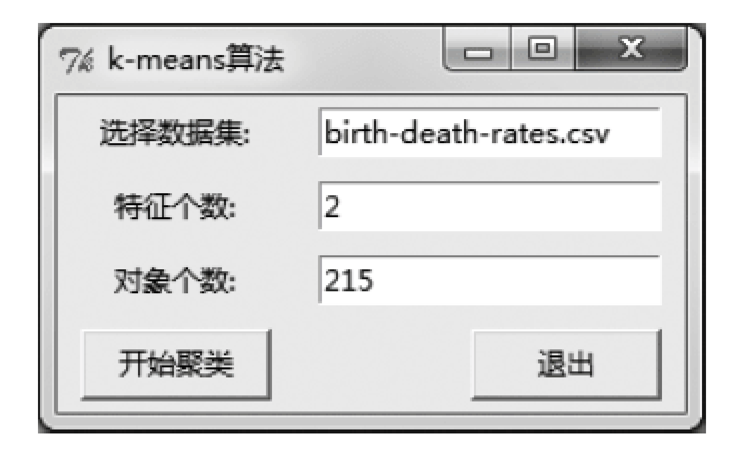


图 6-24　*k*-means 系统的初始界面

单击“开始聚类”按钮开始聚类过程，聚类过程的程序如图 6-25 所示。

```
#聚类过程
def kmenans(dataset, randomArr, n):
#param dataset:所有样本集合
#param randomArr:随机样本
#param n:划分为n个簇
newsortArr=sort(dataset, randomArr, n)
newRandomArr=[]
if len(newRandomArr)==0:
     newRandomArr=randomArr
else:
     for i in range(len(newsortArr)):
          randomArr[i]=pointCenter(newsortArr[i])
     while randomArr!=newRandomArr:
          newRandomArr=randomArr
          newsortArr=sort(dataset, newRandomArr, n)
return newsortArr
#return:划分完以后的簇集合
```

图 6-25　聚类过程程序清单

图 6-25 所示程序清单包含了 k-means 聚类的基本过程，其中 randomArray 函数返回随机样本，sort 函数对数据对象进行类簇划分。

下面看一下实际效果，在 Python 提示符下输入如图 6-26 所示的程序。

```
#实现可视化
if__name__=="__main__":
    startTime=time.clock()
    df=pd.read_csv('birthdeathrates.csv'
    data=df.values[:, 1:]).tolist()#过滤前两列
    for in in range(len(data)):
        for j in range(len(data[0])):
            if type(data[i][j]) is types.Long fype:
                data[i][j]=float(data[i][j])
    #k取3k=3
    randomArr=randomArray(data, k)
    res=kmeans(data, randomArr, k)
    s=0.0
    print(res)
    for i in range(len(res)):
        s+=classInnerDis(res[i])
    sres=classInnerDis(data)
    endTime=time.clock()
    plt.scatter(*zip(*res[0]), c='b')
    plt.scatter(*zip(*res[1]), c='r')
    plt.scatter(*zip(*res[2]), c='g')
    plt.show()
```

图 6-26　可视化程序清单

如图 6-27 所示，根据数据点的分布，大致可将这 215 个国家划分为 3 个区域。从图中可以大致看出，红色区域的国家表现为出生率低、死亡率高，其中包括典型的“少数民族”国家——俄罗斯；绿色区域的国家则是出生率低、死亡率低，这部分国家主要集中在欧洲，包括德国、丹麦、瑞典等国家；相比而言，蓝色区域的国家出生率和死亡率都偏高，这些

国家依然大都出现在非洲。

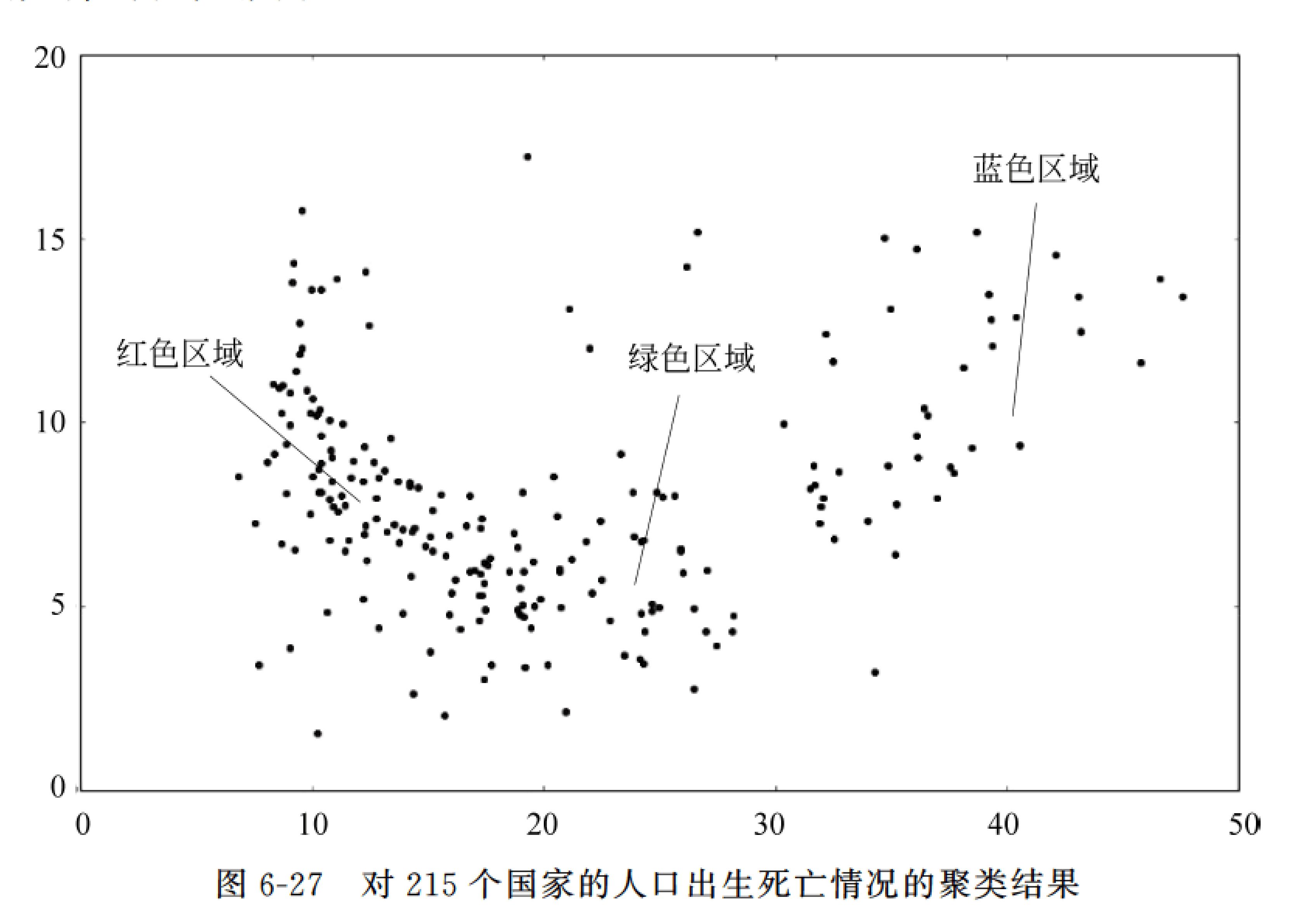


图 6-27　对 215 个国家的人口出生死亡情况的聚类结果

6.8 小　　结

- **簇**是数据对象的集合，同一个簇中的对象彼此相似，而不同簇中的对象彼此相异。将物理或抽象对象的集合划分为相似对象的类的过程称为**聚类**。
- 聚类分析具有广泛的**应用**，包括商务智能、图像模式识别、Web 搜索、生物学和安全。聚类分析可以作为独立的数据挖掘工具来获得对数据分布的了解，也可以作为在检测的簇上运行的其他数据挖掘算法的预处理过程。
- 聚类是数据挖掘研究的一个富有活力的领域。它与机器学习的**无监督学习**有关。
- 聚类是一个充满挑战性的领域，其典型的**要求**包括可伸缩性、处理不同类型的数据和属性的能力、发现任意形状的簇、确定输入参数的最小领域知识需求、处理噪声数据的能力、增量聚类和对输入次序的不敏感性、聚类高维数据的能力、基于约束的能力，以及聚类的可解释性和可用性。
- 目前已经开发了许多聚类算法，这些算法可以从多方面分类，如根据划分标准、簇的分离性、所使用的相似性度量和聚类空间。本章讨论如下几类主要的基本聚类方法：划分方法、层次方法、基于密度的方法和概率模型的聚类方法。
- **划分方法**首先创建 k 个分区的初始结合，其中参数 k 是要构建的分区数。然后，它采用迭代重定位技术，试图通过把对象从一个簇移到另一个簇来改进划分的质量。典型的划分方法包括 k-means、k 中心点、CLAEANS。
- **层次方法**创建给定数据对象集的层次分解。根据层次分解的形成方式，层次方法可以分为凝聚的（自底向上）或分裂的（自顶向下）。典型的层次方法包括 BIRCH、CURE、Chameleon。

- **基于密度的方法**基于密度的概念来聚类对象。一种代表性方法是 DBSCAN，它使用基于中心的方法定义相似度，根据邻域中对象的密度来生成簇。其他典型的基于密度的方法还有 OPTICS、DENCLUE。
- **基于概率模型的聚类**假定每个簇是一个有参分布。使用待聚类的数据作为观测样本，可以估计簇的参数。
- **混合模型**假定观测对象是来自多个概率簇的实例的混合。从概念上讲，每个观测对象都是通过如下方法独立地产生的：首先根据簇概率选择一个概率簇，然后根据选定簇的概率密度函数选择一个样本。
- **期望最大化(EM)算法**是一个框架，它逼近最大似然或统计模型参数的后验概率估计。EM 算法可以用来计算模糊聚类和基于概率模型的聚类。
- **聚类评估**对在数据集上进行聚类分析的可行性和由聚类方法产生的结果的质量进行估计。任务包括评估聚类趋势、确定簇数和测定聚类的质量。

6.9 习　　题

1. 假设数据挖掘的任务是将如下的 8 个点(用(x,y)代表位置)聚类为 3 个簇：

$A_1(2,10)$，$A_2(2,5)$，$A_3(8,4)$，$B_1(5,8)$，$B_2(7,5)$，$B_3(6,4)$，$C_1(1,2)$，$C_2(4,9)$

距离函数是欧氏距离。假设初始时选择 A_1、B_1 和 C_1 分别为每个簇的中心，用 k 均值算法给出：

(1) 在第一轮执行后的 3 个簇中心。

(2) 最后的 3 个簇。

2. k-means 和 k 中心点算法都可以进行有效的聚类。

(1) 概述 k-means 和 k 中心点相比较的优缺点。

(2) 概述这两种方法与层次聚类方法(如 AGNES)相比有何优缺点。

3. 对于 k-means 算法，有趣的是通过小心地选择初始簇中心，或许不仅可以加快算法的收敛速度，而且能够保证结果聚类的质量。k-means++算法是 k-means 算法的变形，它按以下方法选择初始中心。首先，它从数据对象中随机地选择一个中心。迭代地，对于每个未被选为中心的对象 p，选择一个作为新中心。该对象以正比于 $\text{dist}(p)^2$ 的概率随机选取，其中 $\text{dist}(p)$是 p 到已选定的最近中心的距离。迭代过程继续，直到选出 k 个中心。

解释为什么该方法不仅可以加快 k-means 算法的收敛速度，而且能够保证最终聚类结果的质量。

4. 假定一个数据集：

- 有 m 个点，k 个簇。
- 一半的点和簇在“较稠密”区域。
- 一半的点和簇在“不太稠密的”区域。
- 两个区域之间是明显分离的。

对于给定的数据集，下面哪种做法可以最小化寻找 k 个簇时的平方误差？

(1) 在较稠密和不太稠密的区域质心分布应当相同。

(2) 不太稠密的区域应当分配更多的质心。

(3) 较稠密的区域应当分配更多的质心。

5. 有时,层次聚类用来产生 k 个簇,$k>1$。方法是取树状图的第 k 层(根在第一层)的簇。通过观察这种方法产生的簇,可以评估不同数据和簇类型的层次聚类行为,并且将层次聚类与 k-means 进行比较。

下面是一维点的集合:{6,12,18,24,30,42,48}。

(1) 对于下列每组初始质心,将每个点指派到最近的质心,创建两个簇,然后对两个簇的每组质心分别计算总平方误差。对每组质心,给出这两个簇和总平方误差。

① {18,45}; ② {15,40}

(2) 两组质心代表稳定解吗? 即如果在该数据集上使用给定的质心作为初始质心运行 k-means,所产生的簇会有改变吗?

(3) 使用单连接产生的簇是什么?

(4) 在此情况下,哪种技术(k-means 或单连接)能够产生"最自然的"簇?(对于 k 均值,用最小平方误差产生聚类。)

6. 平面上 k 个点集合的 Voronoi 图是将平面上所有点分成 k 个区域的一个划分,使得(平面上)每个点都指派到 k 个指定点中最近的一个(见图 6-28)。Voronoi 图与 k-means 簇之间的关系是什么? 关于 k-means 簇的可能形状,Voronoi 图能告诉我们什么?

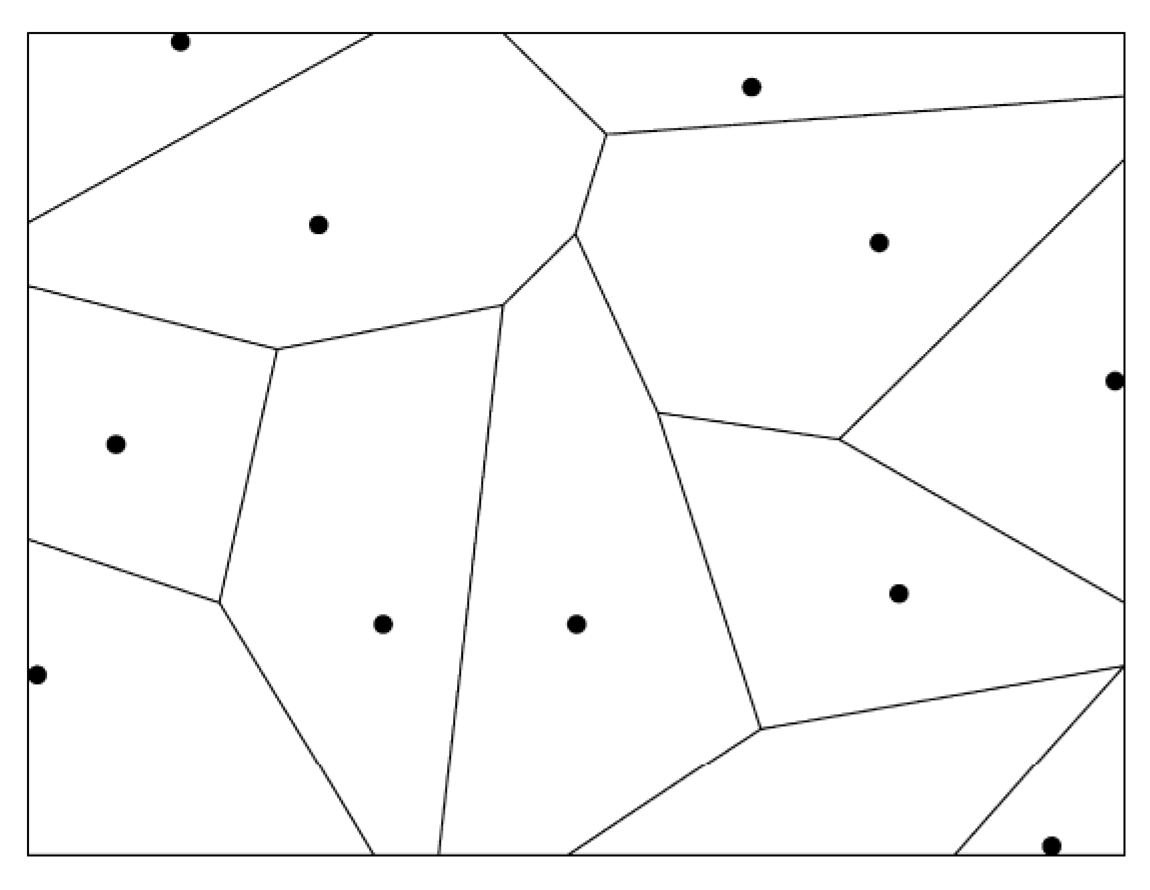


图 6-28 习题 6 的 Voronoi 图

7. 总 SSE 是每个属性的 SSE 之和。如果对于所有的簇,某变量的 SSE 都很低,这意味什么? 如果只对一个簇很低呢? 如果对所有簇都高呢? 如果仅对一个簇高呢? 如何使用每个变量的 SSE 信息改进聚类?

8. 使用图 6-29 中的相似度矩阵进行单连接和全连接层次聚类。绘制树状图显示结果。树状图应当清楚地显示合并的次序。

9. 给定具有 100 个记录的数据集,要求对数据聚类。使用 k-means 对数据聚类,但

	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

图 6-29 习题 8 的相似度矩阵

是对所有的 k 值，k-means 算法都只返回一个非空簇。再用 k-means 的增量版本，但得到的结果完全相同。这是怎么回事？用层次聚类的单连接或 DBSCAN 处理该数据，结果如何？

10. 传统的凝聚层次聚类过程每步合并两个簇。这样的方法能够正确地捕获数据点集的（嵌套的）簇结构吗？如果不能，解释如何对结果进行后处理，以得到簇结构更正确的视图。

11. 从如下 3 个角度对下列每种聚类方法进行描述：①可以确定的簇的形状；②必须指定的输入参数；③局限性。

（1）k-means

（2）k-中心点

（3）CLARA

（4）DBSCAN

12. 指出在何种情况下，基于密度的聚类方法比基于划分的聚类方法和层次聚类方法更适合。并给出一些实例来支持你的观点。

13. 给定两个点集，每个点集包含 100 个落在单位正方形中的点。一个点集这样安排，使得点在空间中均匀地分布。另一个点集由单位正方形上的均匀分布产生。

（1）这两个点集之间有差别吗？

（2）如果有，对于 $k=10$ 个簇，哪一个点集通常具有较小的 SSE？

（3）DBSCAN 在均匀数据集上的行为如何？在随机数据集上呢？

14. 对于 8 个对象{p1，p2，p3，p4，p5，p6，p7，p8}和图 6-30 显示的层次聚类，计算层次 F 度量。类 A 包含点 p1、p2 和 p3，而 p4、p5、p6、p7 和 p8 属于类 B。

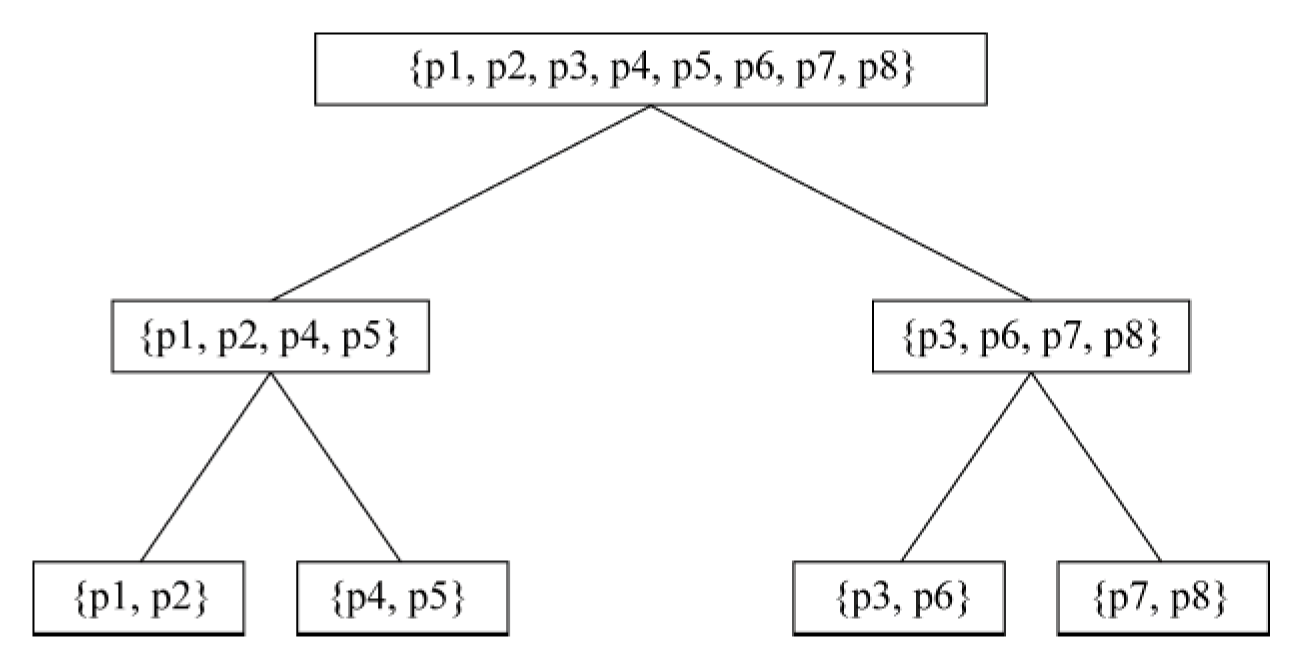


图 6-30 习题 14 的层次聚类

15. 计算表 6-5 中的混淆矩阵的熵和纯度。

表 6-5　习题 15 的混淆矩阵

簇	娱乐	财经	国外	国内	都市	体育	合计
1	1	1	0	4	11	676	693
2	27	89	333	253	827	33	1562
3	326	465	8	16	105	29	949
合计	354	555	341	273	943	738	738

16. 给出一个例子来说明如何集成特定的聚类方法，例如，一种聚类算法被用作另一种算法的预处理步骤。此外，请解释为什么两种聚类方法的集成有时会改进聚类的质量和有效性。

17. 聚类已经被认为是一种具有广泛应用的、重要的数据挖掘任务。对如下每种情况给出一个应用实例：

(1) 把聚类作为主要的数据挖掘功能的应用。

(2) 把聚类作为预处理工具，为其他数据挖掘任务作数据准备的应用。

18. 图 6-31 显示具有两个簇的二维点集的聚类。左边的簇(点用星号标记)有些分散，而右边的簇(点用圆标记)是紧凑的。而紧凑簇的右边有一个单独的点(用箭头指出)属于散开的簇。该簇的中心比紧凑簇的中心远得多。解释为什么用 EM 聚类这是可能的，但是用 k-means 聚类不可能。

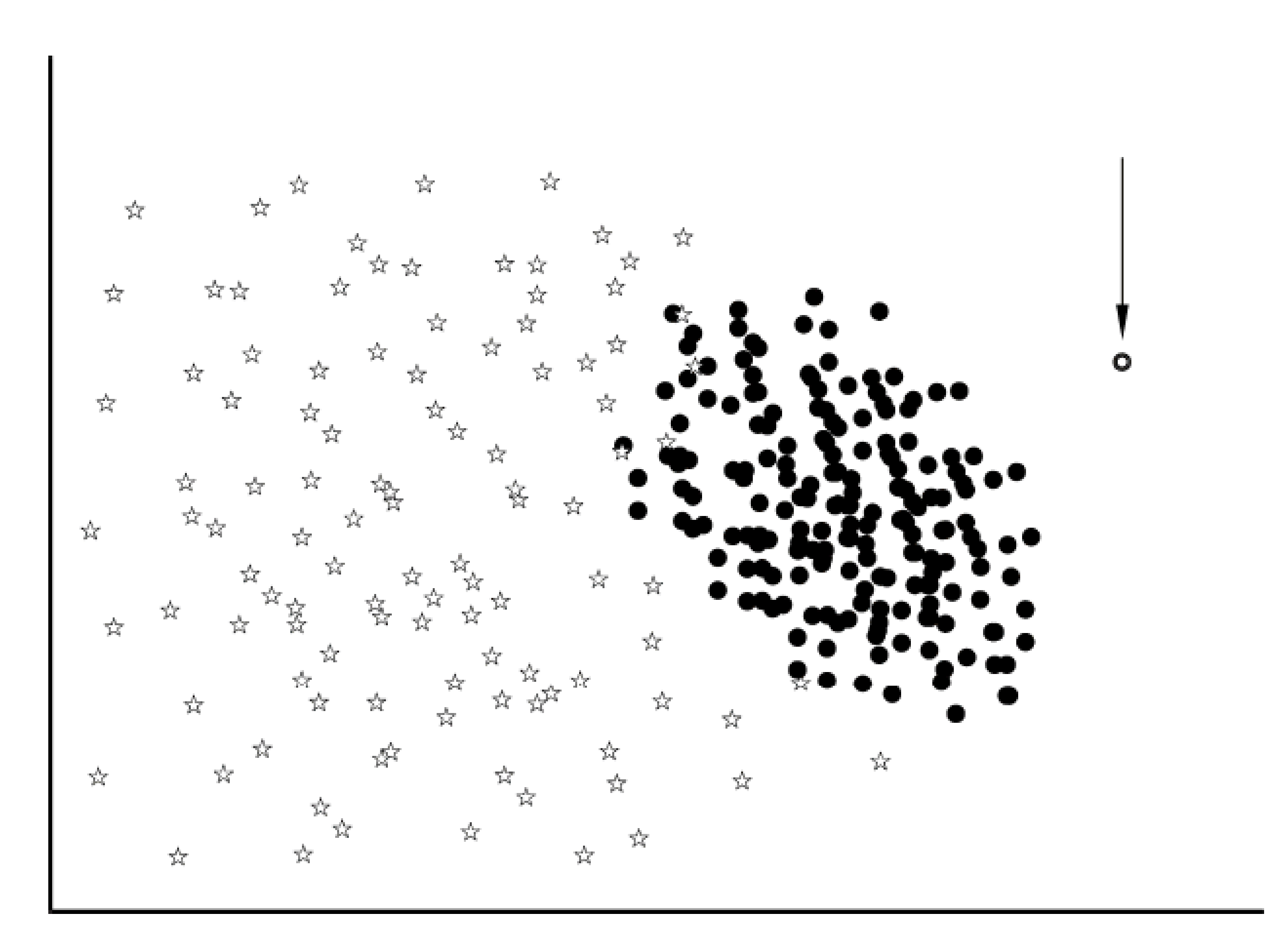


图 6-31　习题 18 的数据集。具有两个不同密度的簇的二维点集的 EM 聚类

19. 传统的聚类方法是僵硬的，应为它们要求每个对象排他性地只属于一个簇。解释为什么这是模糊聚类的特例。你可以使用 k-means 作为例子。

20. 传统的 k-means 具有许多局限性，例如对离群点敏感、难以处理不同大小和不同密度或具有非球形形状的簇。评论模糊 c 均值处理这些问题的能力。

6.10 参考文献

[1] Hartigan J A. Clustering Algorithms. John Wiley&Sons, 1975.

[2] Jain A K, Dubes R C. Algorithms for Clustering Data. Prentics-Hall, 1988.

[3] Kaufman L, Rousseeuw P J. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley&Sons, 1990.

[4] Arabie P, Hubert L J, De Soete G. Clutering and Classification. World Scientific, 1996.

[5] Duda R O, Hart P E, Stork D G. Pattern Classification. 2nd ed. John Wiley&Sons, Inc., New York, 2001.

[6] Mitchell T. Machine Learning. McGraw-Hill, Boston, Ma, 1997.

[7] Hastie T, Tibshirani R, Friedman J H. The Elements of Statistical Learning: Data Mining, Inference, Prediction. Springer, New York, 2001.

[8] Jain A K, Mury M N, Flynn P J. Data Clustering: A Review. ACM Computering Survey,1999,31(3): 264-323.

[9] Han J, Kamber M, Tung A. Spatial Clustering Methods in Data Mining: A Review. In H. J. Miller and J. Han, editors, Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2001: 188-217.

[10] Berkhin P, Merugu S, Dhillon I S, et al. Clustering with Bregman Divergences. In Proc. of the 2004 SIAM Intl. Conf. on Data Mining, Lake Buena Vista, FL, April 2004: 234-245.

[11] MacQueen J. Some Methods for Calssification and Analysis of Multivariate Observations. In Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability. University of Califormia Press, 1967: 281-297.

[12] Anderberg M R. Cluster Analysis for Applications. Academic Press, 1973.

[13] Jain A K, Dubes R C. Algorithms for Clustering Data. Prentice Hall, 1988.

[14] Jardine N, Sibson R. Mathematical Taxonomy. Wiley, 1971.

[15] Sneath P H A, Sokal R R. Numerical Taxonomy. Freeman, 1971.

[16] Zahn C T. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. IEEE Transactions on Computers, 1971,C-20(1): 68-85.

[17] Ester M, Kriegel H P, Sander J, et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proc, of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining. AAAI Press,1996: 226-231.

[18] Sander J, Ester M, Kriegel H P, et al. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. Data Mining and Knowledge Discovery, 1998,2(2): 169-194.

[19] Hoppner F, Klawonn F, Kruse R, et al. Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition. Wiley, 1999.

[20] Bezdek J C. Pattern Recognition with Fuzzy Obdective Function Algorithms. Plenum Press, 1981.

[21] Fraley C, Raftery A E. Model-based Clustering, Discriminant Analysis, and Density Estimation. J. American Statistical Association, 2002,97: 611-631.

[22] Dempster A P, Laird N M, Rubin D B. Maximum Likelihood from Incomplete Data via the EM

Algorithm. J. Royal Statistical Society，1977，Series B，39：1-38.

[23] Halkidi M，Batistakis Y，Vazirgiannis M. On Clustering Validation Techniques. J. Intelligent Information Systems，2001，17：107-145.

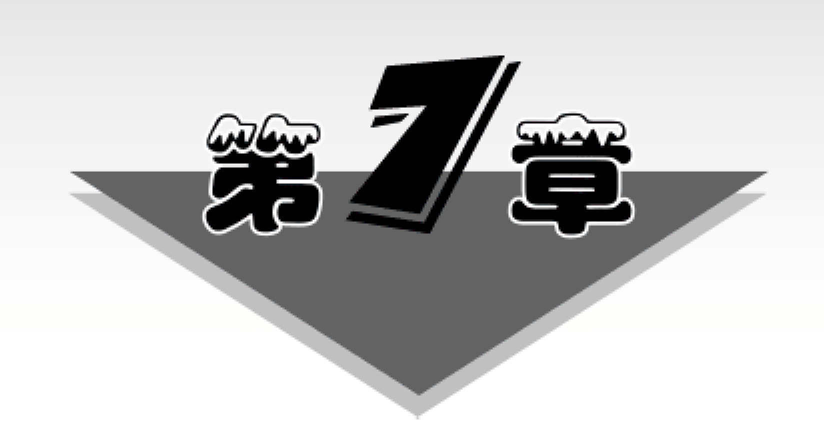
[24] Amigo E，Gonzalo J，Artiles J，et al. A Comparison of Extrinsic Clustering Evaluation Metrics based on Formal Constraints. Information Retrieval，2009，12(4)：461-486.

[25] Halkidi M，Batistakis Y，Vazirgiannis M. Cluster Validity Methods：Part I. SIGMOD Record，2002，31(2)：40-45.

[26] Halkidi M，Batistakis Y，Vazirgiannis M. Clustering Validity Checking Methods：Part Ⅱ. SIGMOD Record，2002，31(3)：19-45.

[27] Milligan G W. Clustering Validation：Results and Implications for Applied Analyses. In P. Arabie，L. Hubert，and G. D. Soete，editors，Clutering and Classification. World Scientific，1996：345-375.

[28] Kleinberg J M. An Impossibiliry Theorem for Clutering. In Proc. of the 16th Annual Conf. on Neural Information Processing Systems，2002：9-14.



深度学习

人类在探索人脑结构与功能、模拟人脑工作机理以及人造大脑的道路上孜孜不倦，沿着这条道路一直走下去，可能永远达不到目标，但发现了一些有意义的东西。人工神经网络就是模拟人脑层次连接结构、大脑神经元处理功能的连接主义方法论指导下的成功模型。狭义深度学习不过是人工神经网络经历懵懂启蒙（神经元模型，1943 年）、不可一世的热潮（感知机，1958 年）、寒冷冰谷（XOR 问题瓶颈，1969 年）、略有起色（BP 算法，1986 年）和走下神坛的复杂问题应用突破（深度学习，2006 年）后又一个春天。自从 20 世纪 80 年代中期到 21 世纪初人工神经网络一直处于浅层神经网络的稳定发展与小规模问题求解，近年来在大数据量和高计算能力驱动下，以深层神经网络求解复杂问题的深度学习开始受到人们的广泛关注，甚至家喻户晓。

深度学习（Deep Learning，DL）是人工智能、机器学习研究中的一个非常有潜力的领域，其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人类大脑的机制来解释数据，例如图像、声音和文本。由于深度学习模型能够在大规模训练数据上取得更好的效果，因此在机器学习领域中有着良好的应用前景。本章从深度学习的发展和基本概念（7.1 节）开始介绍，然后再具体分析深度学习的几种经典模型与算法，包括最常用的深信网（7.2 节）、深玻尔兹曼机（7.3 节）、栈式自动编码器（7.4 节）和卷积神经网络（7.5 节）。本章还简要介绍了几种深度学习开源平台（7.6 节）并给出了一个具体案例，使读者能够熟悉深度学习在实际应用中的整个工作过程。最后，总结了几点深度学习的实用技巧（7.7 节）。

7.1 引　　言

本节首先描述深度学习的发展背景，然后比较深度学习的几种常见的经典模型和它们的相关知识。

7.1.1 发展背景

早在 1943 年，McCulloch 和 Pitts 就提出了一种“MP 神经元模型”来模拟大脑神经元的工作机理，这种模型在人工神经网络以及深度学习中一直沿用至今。在该模型的信号传递过程中，某一待处理的神经元会接收到其他神经元的输入信号，并通过权重的连接进行传递，即加权求和运算获得神经元输入，该神经元接收到的总输入值与其阈值进行比较，并通过“激活函数”（activation function）处理，最后产生神经元的输出。MP 模型很容

易实现与、或、非运算。但是,随着后来某些现实任务的复杂,人们发现其学习能力非常有限,甚至不能解决简单的“异或”问题。直到在20世纪80年代,误差反向传播(Back Propagation, BP)训练算法开始发展起来,研究者发现利用BP算法可以让一个人工神经网络模型从大量训练样本中学习输入输出的复杂非线性映射关系,从而对未知事件做预测,由此掀起了神经网络第二次高潮。这使得神经网络再度引起广泛关注,BP算法也成为此后几十年来神经网络的应用和研究中最为典型的学习算法。BP算法的发明对深度学习的提出与发展做出了巨大贡献,在当时基于BP算法的这样一些神经网络也被广泛称作为多层感知机(Multi-Layer Perceptron, MLP)。20世纪90年代,研究者发现从样本集中学习统计规律能使基于统计的机器学习方法比起过去基于人工规则的系统在很多方面显出优越性,一种影响巨大的浅层机器学习模型被提出,即支持向量机(Support Vector Machines, SVM),这种模型的结构基本上可以看成带有单层隐层节点的神经网络,且在理论分析和应用中都获得了较大的成功。相比之下,典型的浅层人工神经网络拟合输入输出的非线性映射关系,而中间隐层类似于黑箱,使得理论分析的难度大,训练方法又需要较多经验和技巧,使得这个时期浅层人工神经网络相对沉寂。

2006年,加拿大多伦多大学教授Hinton等人提出了深度学习,并表达了深度学习的两种基本特点:第一,多隐层的人工神经网络具有优异的特征学习能力,学习得到的特征对数据有更本质的刻画,从而有利于对原始数据进行识别或分类;第二,深度神经网络在训练上的难度可以通过“逐层初始化预训练”(layer-wise pretraining)来有效降低。此后,以深度学习为代表的复杂模型便开始受到人们的关注,并迎来了神经网络的又一次浪潮。

7.1.2 基本概念

经过近十年的发展,深度学习基于不同的结构和原理已经涌现出多种经典且实用的模型。典型的深度学习模型有深信网、深度波尔兹曼机、栈式自动编码器、卷积神经网络等,表7-1和表7-2描述了深度学习的这几种模型并比较了它们之间的异同。

表7-1 深度学习的几种典型模型

模型	深信网 DBN	深度玻尔兹曼机 DBM	栈式自动编码器 SAE	卷积神经网络 CNN
思想	层层堆叠多个RBM组成的深网络,逐层贪婪预训练,每一层学习过程即为RBM训练过程。预训练完成后,将网络展开为深层次前向网络,再运用BP算法进行微调	借鉴能量模型RBM的基本思想,通过增加RBM的隐层数量而构造的深网络,学习过程同RBM相似,但层次增多	通过组合多个“自动编码”网络(类似于三层自学习BP网络)而构造的深网络,首先逐层贪婪预训练,每一层学习过程同BP算法一致。预训练完成后,将网络展开为深层次前向网络,再运用BP算法进行整体微调	针对二维数据设计的一种模拟“感受野”的功能结构,通过多次卷积和池化过程构造的深网络,网络的训练含有“权共享”和“稀疏”特点,学习参数过程类似于BP算法

续表

模型	深信网 DBN	深度玻尔兹曼机 DBM	栈式自动编码器 SAE	卷积神经网络 CNN
异同点	DBN 和 DBM 的相同点在于二者都是基于能量模型 RBM 的改进，预训练过程都与 RBM 的原理相似；不同在于 DBN 由多个 RBM 组成，而 DBM 可以看作一个多隐层的 RBM		SAE 和 CNN 的相同点在于：模型训练中，误差反向传播过程都与 BP 算法相似；不同在于：SAE 每层之间全连接，CNN 每层之间进行局部连接，每次进行小块的二维卷积和池化	
训练模式	逐层贪婪预训练和整体网络微调	整体网络训练	逐层贪婪预训练和整体网络微调	整体网络训练
有监督/无监督	无监督预训练 监督微调	无监督	无监督	有监督
优点	对数据进行更好的特征学习，训练方便	更好的学习数据的深层隐特征，有利于分类或可视化	解决无标签数据的特征学习过程，在模式识别方面应用广泛	学习能力强，特征提取方面效果很好，分类正确率高
缺点	每一层的训练受上一层的训练结果影响，容易过拟合	深层网络统一训练，参数更新缓慢，复杂度高	容易导致过拟合，且具有一定的复杂度	调参比较麻烦，网络有一定的复杂度，训练缓慢

表 7-2　深度学习几种模型的目标函数对比

模　型	目 标 函 数
深信网	预训练目标函数：$E(v,h)=-\sum_{i=1}^{m} v_i b_i-\sum_{j=1}^{n} h_j c_j-\sum_{i=1,j=1}^{m,n} v_i h_j w_{ij}$ 微调目标函数：$E(w,b)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\parallel \hat{x}_i-x_i\parallel^2$
深度玻尔兹曼机	$E(v,h;\Psi)=-\sum_{ij} w_{ij}^{(1)} v_i h_j^{(1)}-\sum_{k=2}\sum_{jl} w_{ij}^{(k)} h_j^{(k-1)} h_l^{(k)}$
栈式自动编码器	$E(w,b)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\parallel \hat{x}_i-x_i\parallel^2$
卷积神经网络	$E(w,b)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\parallel \hat{y}_i-y_i\parallel^2$

总的来讲，深度学习可以简单理解为深层神经网络的一种新的学习范式，有着更好的学习能力，它是通过组合低层特征形成更加抽象的高层表示，以此更好地发现原始数据的属性类别或特征表达。深度学习一定程度上改善了以往传统神经网络在训练中所表现出来的目标函数优化的局部极小、不能收敛到稳定状态等问题，并且在高性能计算平台支撑下对图像、语音、文本数据的理解、识别等复杂问题求解取得了巨大突破。下面对几种典型的深度学习模型与算法展开具体分析。

7.2　深　信　网

深度学习取得成功的第一个典型深层次结构模型就是深信网(Deep Belief Network，DBN)，它是由多个受限玻尔兹曼机堆叠而成的网络。其训练过程包括预训练和微调两

个阶段，预训练阶段是一种通过多个独立受限玻尔兹曼机的无监督逐层训练，上一个受限玻尔兹曼机训练成熟后的输出直接作为下一个受限波尔兹曼机的输入；微调阶段是将链接起来的深层前向神经网络通过 BP 算法进行微调校正权值。接下来本节首先对玻尔兹曼机结构进行介绍，然后介绍受限的玻尔兹曼机，最后描述深信网的详细结构和算法。

7.2.1 玻尔兹曼机

在神经网络大家族中定义了一种层间全连接层内互联的网络模型，并为网络状态定义了一个“全局能量”，在这种能量最稳定即能量最小的时候，该网络就达到理想状态。对该网络的训练即为最小化这个能量函数，这样的模型称为玻尔兹曼机（Boltzmann Machine，BM），其简单结构如图 7-1 所示，在玻尔兹曼网络中每一个神经元的取值都是布尔型的，即只能够取 0 和 1 两种状态，为 1 表示激活，为 0 则表示抑制。

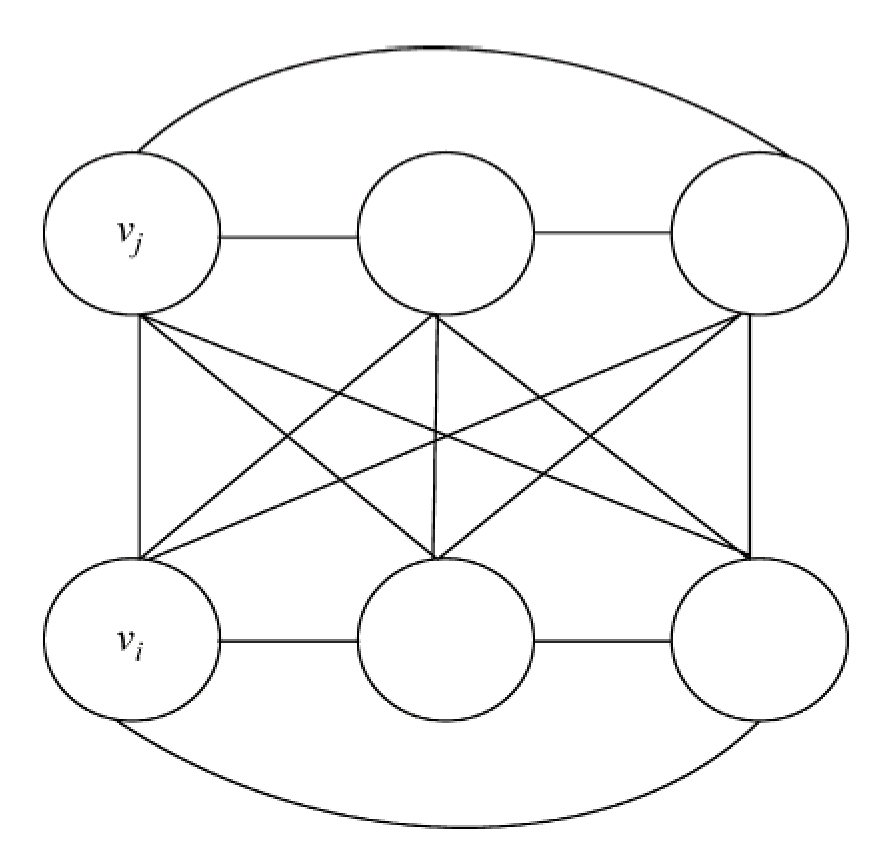


图 7-1 玻尔兹曼机结构

图中，v_i 和 v_j 表示网络中第 i 和 j 个神经元节点状态值，w_{ij} 表示这两个神经元节点之间的连接权，θ_i 表示第 i 个神经元节点阈值，在 Hebb 学习规则启发下，该玻尔兹曼机网络的联合能量函数可以定义为

$$E(v) = -\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}v_iv_j - \sum_{i=1}^{n} v_i\theta_i \tag{7-1}$$

网络训练过程就是通过最小化能量函数，将每个训练样本与隐神经元状态整体视为一个玻尔兹曼机网络的当前状态向量，使其出现的概率尽可能大，利用联合概率分布的最大似然函数可以更新网络的连接权。联合概率模型的一种合理策略是神经元状态的联合概率与它们的能量函数成反比例，利用指数簇分布可以定义网络某一状态向量 $\boldsymbol{v}$ 出现的概率，即由其能量与所有可能状态向量 $\boldsymbol{t}$ 的能量和（满足全概公式）来确定：

$$P(\boldsymbol{v}) = \frac{e^{-E(\boldsymbol{v})}}{\sum_t e^{-E(\boldsymbol{t})}} \tag{7-2}$$

7.2.2 受限玻尔兹曼机

标准玻尔兹曼机的总体结构是一个全连接图，其整体网络的训练复杂度极高，对于一些复杂数据并不能很好地处理。在现实任务中，通过限制网络结构为两层结构且层内没有互连，提出了受限玻尔兹曼机（Restricted Boltzmann Machine，RBM）。与 BM 不同在于，RBM 模型的节点连接是一个二分图，它含有随机二进制单元的两层网络，分别称为可视层和隐层，两层之间是全连接且带有对称的权重，而每一层内部节点之间无连接。可视层单元是一些可观察的实值数据（通常约束为 0/1 状态值），隐层节点值取 0/1 状态，代表

数据的隐特征层。RBM 的基本结构如图 7-2 所示。

在 RBM 中，输入层 v 和隐层 h 的联合能量函数定义如下：

$$E(v,h) = -\sum_{i=1}^{m} v_i b_i - \sum_{j=1}^{n} h_j c_j - \sum_{i=1,j=1}^{m,n} v_i h_j w_{ij} \tag{7-3}$$

其中，v_i 和 h_j 是可视层和隐层的网络节点值，b_i 和 c_j 是二者对应节点阈值，w_{ij} 是连接两层节点之间的连接权。

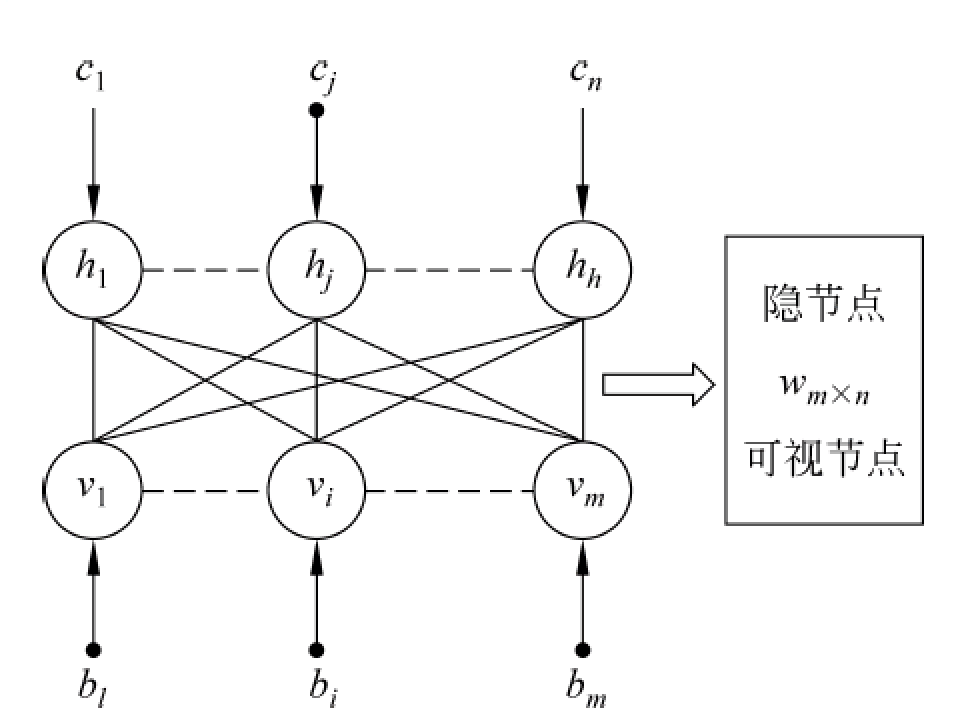


图 7-2 受限玻尔兹曼机结构

通过能量函数为可视层每一个节点对应状态随机变量定义了抽样时的概率密度函数如下：

$$P(v) = \frac{\sum_{h} e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}} \tag{7-4}$$

其中分母表示一种“配分函数”形式，代表了可视层和隐层所有节点的一种基于能量的活跃度，能量越小，则某一节点值越活跃；分子代表了基于所有隐层能量得到的可视层节点激活值。

从式(7-4)可以看出，为了使得可视层某一节点 v 的当前概率 $p(v)$ 最大化，则需要降低当前节点的能量，并且同时提高其他节点的能量。通过对式(7-4)的似然函数 $\log p(v)$ 最大化，结合随机梯度上升策略来求得可视层概率极大值，根据梯度上升可获得相应的权值修改量(其中 η 为学习率)：

$$\Delta w_{ij} = \eta \frac{\partial \log p(v)}{\partial w_{ij}} = \eta(< v_i h_j >_{\text{data}} - < v_i h_j >_{\text{model}}) \tag{7-5}$$

RBM 在训练过程中，基于可视层输入 $\boldsymbol{v}$，可以得到第 $\boldsymbol{j}$ 个隐层节点状态更新为激活值 1 的条件概率如下：

$$p(h_j = 1 \mid v) = f\left(\sum_{i=1}^{m} v_i w_{ij} + c_j\right) \tag{7-6}$$

同理，基于隐层节点值 h，也可以得到可视层第 i 节点状态更新为激活值 1 的条件概率如下：

$$p(v_i = 1 \mid h) = f\left(\sum_{j=1}^{n} h_j w_{ij} + b_i\right) \tag{7-7}$$

通常，RBM 的两层网络状态可以从可视层的输入状态值通过式(7-6)和式(7-7)重复交替更新隐层和可视层，这个过程称为“交替执行 Gibbs 抽样”，直到网络获得输入样本的稳定联合分布，网络的输出激活函数通常采用最常用的 Sigmoid 函数：$f(x)=1/(1+e^{-x})$。由输入样本数据作为最开始的可视层状态，已知输入状态条件下执行 Gibbs 抽样产生隐层状态，对应式(7-5)右边第一项；在当前网络连接权保持不变的情况下，RBM 模型经过自底向上-自顶向下的多次 Gibbs 抽样后，抽样产生的可视层状态和隐层状态作为

模型生成的状态，对应式(7-5)右边第二项。这种方式近似获得式(7-5)的模型参数更新公式，同理可以计算阈值的更新变化量(其中 η 为学习率)，即

$$\begin{cases}\Delta w_{ij}=\eta(<v_i h_j>_{\text{data}}-<v_i h_j>_{\text{model}})\\ \Delta b_i=\eta(v_{i\text{data}}-v_{i\text{model}})\\ \Delta c_j=\eta(h_{j\text{data}}-h_{j\text{model}})\end{cases}\tag{7-8}$$

因此网络训练时连接权值和阈值的更新迭代公式为

$$\begin{cases}w_{ij}\leftarrow w_{ij}+\Delta w_{ij}\\ b_i\leftarrow b_i+\Delta b_i\\ c_j\leftarrow c_j+\Delta c_j\end{cases}\tag{7-9}$$

7.2.3 深信网

深信网是一个典型的深层神经网络模型，它的网络结构如图 7-3 所示，其模型构造和训练过程分为两阶段：第一阶段是通过 RBM 逐层初始化“预训练”(图 7-3(a))，将每一个 RBM 学习后的隐层输出特征表达直接作为下一个 RBM 两层结构的输入，层与层之间独立按照 RBM 所要求的迭代次数依次进行无监督学习和更新参数；第二阶段是全局“微调”，将预训练阶段的多个 RBM 展开链接起来的深层前向神经网络(图 7-3(b))，下半部分称为编码，上半部分称为解码，这样构造的深网络的初始权值就是预训练阶段的一系列 RBM 的权值。由于逐层训练 RBM 结构对数据的特征学习并不是最好的，还需通过 BP

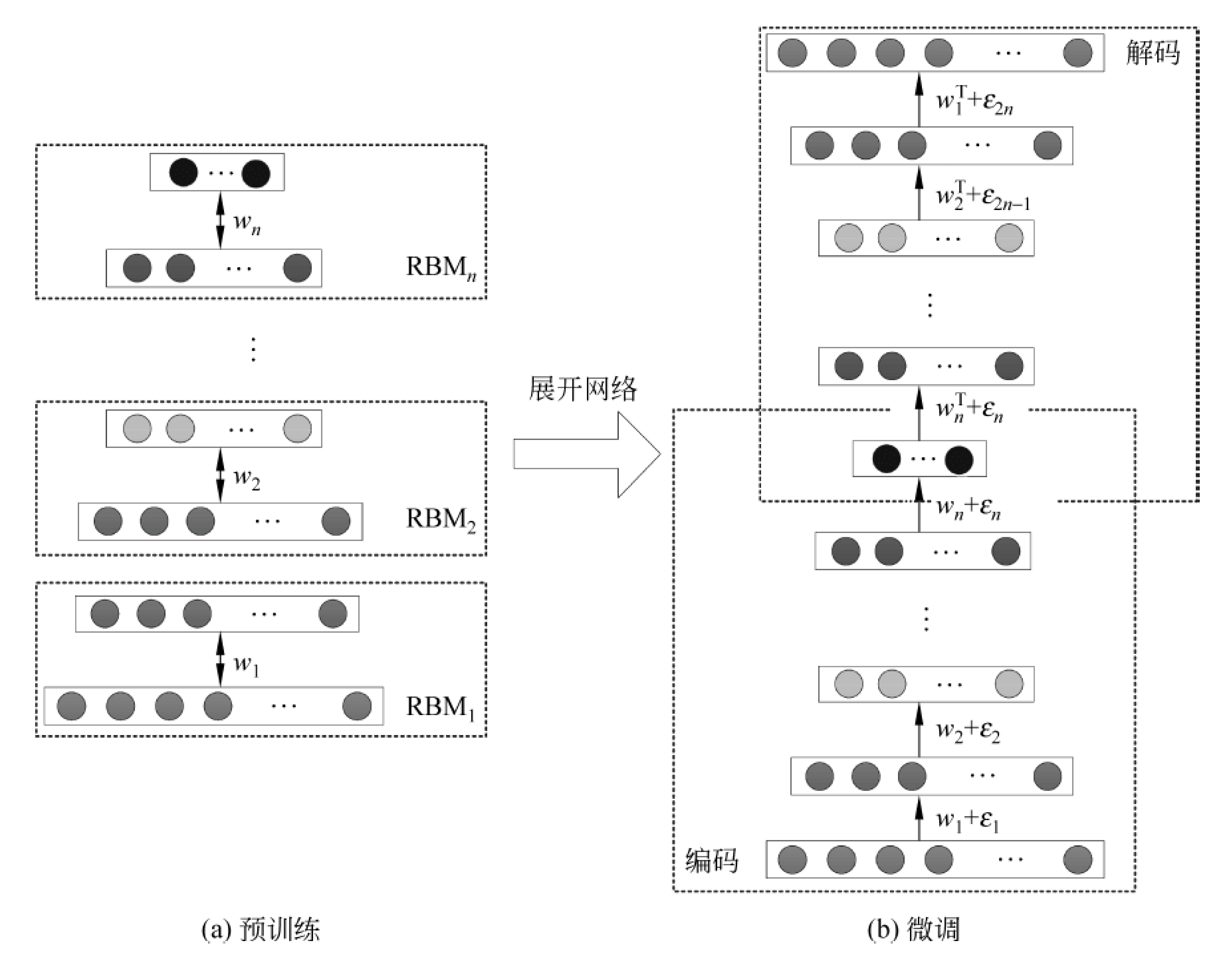


图 7-3　深信网结构

算法进行自监督(输入输出误差最小)的微调校正权值,最终网络结构可以实现维数约简,即通过限制中间神经元的更小个数起到降维的作用。深信网也能用于分类问题,只需在编码部分的顶部加一个分类层,一般是加一个线性分类层,然后只需要误差反向传递修改分类层的权值即可。

从图 7-3 中可以看出,DBN 首先会将大量参数分组,通过预训练从输入层开始逐层为每组找到它们局部最好权的初始设置,然后再基于这些局部较优的结果联合起来构建整体深网络,并采用随机梯度进行全局微调。其预训练阶段被 Hinton 称为非监督逐层贪婪学习,学习规则即为逐层 RBM 的训练过程,直到模型达到稳定状态;在 DBN 微调阶段采用传统多层 BP 神经网络的随机梯度下降法训练,该过程又分为编码学习和解码重构:编码学习就是通过展开后的网络将原始高维数据映射至低维空间,解码重构就是通过相逆的解码网络来重构原始高维数据。算法 7-1 描述了 DBN 预训练过程。

算法 7-1 深信网预训练过程

输入:训练集$V=\{v_1, v_2, \cdots, v_m\}$,网络学习率 η,RBM 个数 N,迭代次数 iter

输出:训练完成的多个 RBM 展开成的初始深网络

步骤:

1: **for** $n=1 \rightarrow N$ **do**
2: 　在(0,1)范围内随机初始化当前 RBM 的连接权值 w_{ij} 和阈值 b_i、c_j
3: 　**repeat**
4: 　　**for** all $v_i \in V$ **do**
5: 　　　根据当前参数和式(7-6)抽样隐层节点状态值 h_j,并结合输入值 v_i 和隐层值 h_j,对应乘积获得当前 RBM 网络的初始联合状态$\langle v_i h_j \rangle_{data}$
6: 　　　根据 h_j 和式(7-7)抽样重构的可视层节点状态值$\hat{v}_i$,并根据$\hat{v}_i$ 和式(7-6)再次抽样重构的隐层节点状态值$\hat{h}_j$
7: 　　　计算$\hat{v}_i$ 和$\hat{h}$的乘积,获得当前 RBM 网络的模型联合状态$\langle v_i h_j \rangle_{model}$
8: 　　　$v_{idata} \leftarrow v_i, h_{jdata} \leftarrow h_j, v_{imodel} \leftarrow \tilde{v}_i, h_{jmodel} \leftarrow \tilde{h}_j$
9: 　　　根据式(7-8)和式(7-9)更新当前 RBM 的连接权值 w_{ij} 和阈值 b_i、c_j
10: 　　**end for**
11: 　**until** 网络训练达到迭代数 iter
12: **end for**

DBN 的微调阶段所用到的基本思想即为 BP 算法原理,目标函数即为原始数据和重构数据的误差。下面详细介绍 BP 算法流程。

BP 算法每次调整权值包括信号正向传播与误差反向传播两个过程。首先样本输入信号从输入层经各隐层神经元逐层由低层向高层传播,最终传向输出层,计算所有样本输入信号对网络的输出与期望样本输出的全局误差;然后检验全局误差,若全局误差未达到要求,则通过误差反向传播由输出层开始逐层修改各层神经元间的连接权。经过若干次信号正向传播与误差反向传播过程,最终使网络对全体样本信号的输出值与期望值的全局误差达到要求。

设第 $m-1$ 层第 i 个神经元与第 m 层第 j 个神经元间的连接权为 $w_{ij}^{(m)}$,第 m 层第 j

个神经元的输入记为 $\text{net}_j^{(m)}$，输出记为 $O_j^{(m)}$，P 个样本中第 p 个样本信号记为(X^p,Y^p)，其中样本序号 $p=1,2,\cdots,P$。下面分析具体过程。

(1) 信号正向传播。

从输入层节点逐层向前计算每一个节点的输入输出，第 j 个节点的输入输出为

$$\text{net}_j^{(m)} = \sum_i w_{ij}^{(m)} O_i^{(m-1)} \tag{7-10}$$

$$O_j^m = f(\text{net}_j^{(m)}) \tag{7-11}$$

其中，当 $m=1$ 时，若输入信号 $x^p=\{x_1^p,x_2^p,\cdots,x_n^p\}$，则 $\text{net}_j^{(1)}=\sum_i w_{ij}^{(1)}x_i^p$；当 m 为输出层时，网络第 k 个节点的输出为：$\hat{Y}_k^p=O_k^{(m)}=f(\text{net}_k^{(m)})$。最后计算训练样本的网络输出$\hat{Y}=\{\hat{y}_1^p,\cdots,\hat{y}_k^p,\cdots,\hat{y}_K^p\}$与相应的理想输出 $Y=\{y_1^p,\cdots,y_k^p,\cdots,y_K^p\}$的误差限为

$$E^P = \frac{1}{2}\sum_{k=1}^{K}(y_k^p-\hat{y}_k^p)^2 \tag{7-12}$$

(2) 反向传播过程。

从输出层向输入层反向进行误差修正，按极小化误差的方式调整权值，控制要求常为每一 $E^P\left(\text{或} E=\sum_p^P E^p\right)$ 满足给定的精度要求。此阶段之所以称为反向传播阶段(或称误差反向传播阶段)，是对应输入信号的正向传播而言的。因为在开始调整神经元的权值时，只能先求输出层的误差，而其他层的误差要通过此误差反向逐层向后推才能得到，因而得名。

(3) 算法分析。

BP 算法过程可以分为两种，一种是按样本顺序提交，另一种则是消除样本顺序影响。

① 按样本顺序提交的 BP 算法。

对样本集$\{(X^1,Y^1),(X^2,Y^2),\cdots,(X^p,Y^p),\cdots,(X^P,Y^P)\}$，网络根据$(X^1,Y^1)$计算出实际输出$\hat{Y}^1$ 和误差测度 E^1，对各层 $w_{ij}^{(m)}$ 做一次调整；在此基础上，再根据(X^2,Y^2)计算实际输出$\hat{Y}^2$ 和误差测度 E^2，再对各层 $w_{ij}^{(m)}$ 做一次调整……如此下去，直到本次循环最后一个样本(X^P,Y^P)计算实际输出$\hat{Y}^P$ 和误差测度 E^P，对各层 $w_{ij}^{(m)}$ 做第 P 次调整。这个过程，相当于是对样本集中各个样本的一次循环处理。这个循环需要重复下去，直到对整个样本集来说误差测度的总和满足系统的要求为止，即 $E<\varepsilon$(此处 ε 为精度控制参数)。

② 消除样本顺序影响的批处理的 BP 算法。

按①所述的算法能在一定程度上抽取出样本集中所含的输入向量和输出向量之间的关系。但是 BP 网络接受样本的顺序仍然对训练的结果有较大的影响。比较而言，网络更"偏向"较后出现的样本：如果每次循环都按$(X^1,Y^1),(X^2,Y^2),\cdots,(X^p,Y^p),\cdots,(X^P,Y^P)$的顺序进行训练，在网络"学成"投入运行后，对于与该样本序列较后的样本较接近的输入，网络所给出的输出的精度将明显高于与样本序列较前的样本较接近的输入对应的输出的精度。那么，是否可以根据样本集的具体情况，给样本集中的样本安排一个适当的顺序，以求达到基本消除样本顺序的影响，获得更好的学习效果呢？这是非常困难的。因

为无论如何排列这些样本，它终归要有一个顺序，序列排得好，顺序的影响只会稍小一些。另外，要想给样本数据排列一个顺序，本来就不是一件容易的事情，再加上要考虑网络本身的因素，就更困难了。

造成样本顺序对结果产生严重影响的原因是：算法对各层 $w_{ij}^{(m)}$ 的调整是分别依次根据(X^1,Y^1)，(X^2,Y^2)，…，(X^p,Y^p)，…，(X^P,Y^P)完成的。“分别”和“依次”决定了网络对“后来者”的“偏爱”。实际上，按照这种方法进行训练，有时甚至会引起训练过程的严重抖动，它可能使网络难以达到用户要求的训练精度。这是因为排在较前的样本对网络的影响被排在较后的样本的影响掩盖了，从而使得排在较后的样本对最终结果的影响就要比排在较前的样本的影响大。

虽然在精度要求不高的情况下，顺序的影响有时是可以忽略的，但是还应该尽量消除它。那么，如何消除样本顺序对结果的影响呢？根据上述分析，算法应该避免“分别”“依次”的出现。因此，不再分别依次根据(X^1,Y^1)，(X^2,Y^2)，…，(X^p,Y^p)，…，(X^P,Y^P)对各层 $w_{ij}^{(m)}$ 进行调整，而是用样本集$\{(X^1,Y^1),(X^2,Y^2),\cdots,(X^p,Y^p),\cdots,(X^P,Y^P)\}$的“总效果”去实施对 $w_{ij}^{(m)}$ 的修改，这就可以较好地将样本集的样本的一系列学习变成对整个样本的学习。

③ 随机梯度下降法。

DBN 微调阶段的训练过程使用的是基于随机梯度下降策略的 BP 算法，该过程每一次执行都会随机选择训练集的一个子集构建目标来求解。即 DBN 在微调过程的训练中会对样本划分进行批次训练，每一次循环都是在分批样本中随机选择一批进行训练，这不仅体现了上述 BP 算法中所描述的消除样本顺序影响，还体现了对大数据分批训练的思想，有利于网络学习到更好的特征。

当计算了一个随机选择的训练样本子集的网络实际输出与期望输出的误差 E，从输出层到第一层按最陡梯度下降法调整连接权，则获得随机选择的训练样本子集“总效果”的最简单的权值更新采取如下通用迭代公式：

$$\begin{cases} w_{ij}^{(m)} \leftarrow w_{ij}^{(m)} - \eta \dfrac{\partial E}{\partial w_{ij}^{(m)}} \\ \dfrac{\partial E}{\partial w_{ij}^{(m)}} = \sum\limits_{p=1}^{P} \dfrac{\partial E^p}{\partial w_{ij}^{(m)}} \end{cases} \tag{7-13}$$

上式中 $\eta(>0)$为迭代步长（学习率），对于$\dfrac{\partial E^P}{\partial w_{ij}^{(m)}}$的具体表达式由于其采用的激励函数而有所变化，形式上是按传统 BP 算法的误差反向传播来计算。算法 7-2 描述了 DBN 微调过程。

算法 7-2 深信网微调过程

输入：训练集 $X=\{x_1,x_2,\cdots,x_m\}$，网络学习率 η，每批训练样本数 batch_size，迭代次数 iter

输出：训练完成的深信网

步骤：

1：根据算法 7-1 进行多个独立的 RBM 预训练，并展开成深层前向 BP 神经网络

2：加载预训练完成的网络连接权值 w_{ij} 和每一层节点阈值 b_i，将 w_{ij} 作为网络编码阶段的初

```
    始连接权值,并将其转置 w_ij^T 作为网络解码阶段的初始连接权值 V_ij
repeat
    将整个训练集 X 按 batch_size 大小随机划分成 N 批
    for n =1→N do
        for all X_n⊂X do
            根据式(7-10)和式(7-11)计算当前批次样本的网络前向输出 X̂_n
            根据式(7-12)计算当前批次样本的输入 X_n 和输出 X̂_n 的误差 E
            根据式(7-13)更新网络的连接权值 w_ij 和阈值 b_i
        end for
    end for
until 网络训练达到迭代数 iter 或所有误差 E 小于给定最大误差界
```

7.3 深玻尔兹曼机

2013 年,R. Salakhutdinov 等人通过仔细研究 RBM 的机理,把某一个 RBM 隐藏层的层数增加,构成了另一种深度学习模型,称为深玻尔兹曼机(Deep Boltzmann Machine,DBM)模型。下面详细描述该模型的结构和算法:

DBM 模型总体结构如图 7-4 所示,DBM 是多层对称连接的随机二进制单元,它包含可视层 $v\in\{0,1\}$ 和 n 个隐层 $h^{(i)}\in\{0,1\}$,模型的任意邻接的两层之间是全连接且带有对称的权重,而每一层内部节点之间无连接。

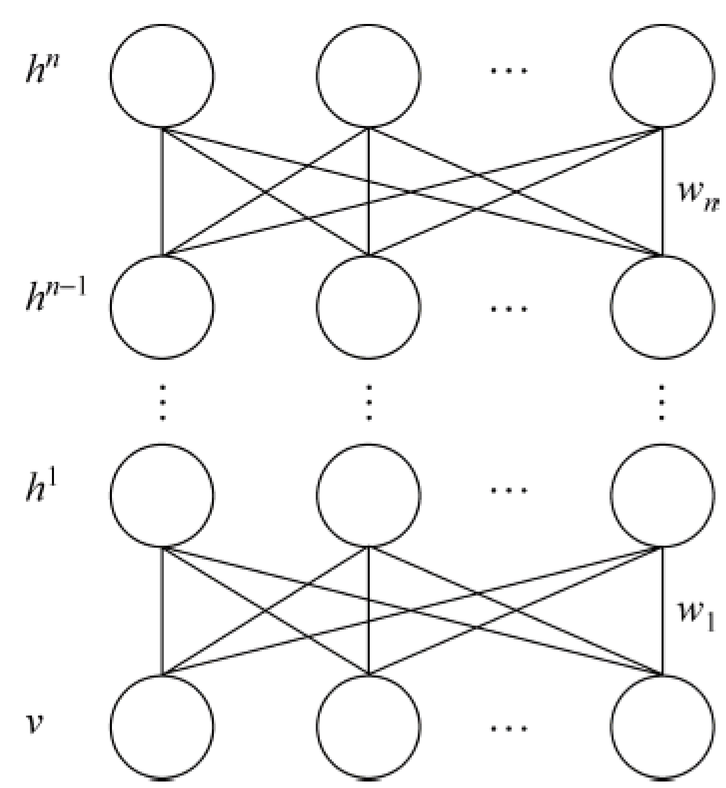


图 7-4 深度玻尔兹曼机结构

DBN 在预训练阶段,只是利用基本的两层 RBM 原理逐层贪婪训练。DBM 和它的最大区别在于 DBM 的某一层节点分布同时依赖其他所有层。下面以可视层 v 和 3 个隐层 $h^{(1)}$、$h^{(2)}$、$h^{(3)}$ 的深玻尔兹曼机(DBM)为例,为了简化并约束每层的阈值固定为 0,则 DBM 网络整体结构$\{v,h\}$的联合能量函数定义如下:

$$E(v,h;\Psi)=-\sum_{ij}w_{ij}^{(1)}v_ih_j^{(1)}-\sum_{jl}w_{ij}^{(2)}h_j^{(1)}h_l^{(2)}-\sum_{lk}w_{lk}^{(3)}h_l^{(2)}h_k^{(3)} \tag{7-14}$$

其中 $h=\{h^{(1)},h^{(2)},h^{(3)}\}$,$\Psi=\{w^{(1)},w^{(2)},w^{(3)}\}$

与 RBM 中的式(7-4)同理,DBM 利用所有隐层节点状态,通过玻尔兹曼分布原理为可视层 v 产生状态的概率分布如下:

$$p(v,\psi)=\frac{1}{z(\psi)}\sum_h\exp(-E(v,h^{(1)},h^{(2)},h^{(3)};\psi)) \tag{7-15}$$

DBM 计算可视层 v 状态的概率分布需要依赖隐层 $h^{(1)}$ 的状态分布,而隐层 $h^{(1)}$ 依赖隐层 $h^{(2)}$ 的状态分布和可视层 v 状态分布,$h^{(2)}$ 依赖隐层 $h^{(1)}$ 同时依赖隐层 $h^{(3)}$。DBM 的中间层的神经元状态的抽样概率依赖于前一层和后一层神经元状态,最高隐层 $h^{(n)}$ 神经元状态的抽样概率只依赖于 $h^{(n-1)}$ 神经元状态。所以 DBM 可以理解成一个多隐层的

RBM，其结构和 RBM 类似。训练过程中，所有层的节点产生激活值 1 的条件概率可通过如下公式计算：

$$\begin{cases} p(h_j^{(1)}=1 \parallel v,h^{(2)}) = f\left(\sum_{i=1}^{D} w_{ij}^{(1)} v_i + \sum_{l=1}^{F_2} w_{jl}^{(2)} h_l^{(2)}\right) \\ p(h_l^{(2)}=1 \parallel h^{(1)},h^{(3)}) = f\left(\sum_{j=1}^{F_1} w_{jl}^{(2)} h_j^{(1)} + \sum_{k=1}^{F_3} w_{lk}^{(3)} h_k^{(3)}\right) \\ p(h_k^{(3)}=1 \parallel h^{(2)}) = f\left(\sum_{l=1}^{F_2} w_{lk}^{(3)} h_l^{(2)}\right) \\ p(v_i=1 \parallel h^{(1)}) = f\left(\sum_{j=1}^{F_1} w_{ij}^{(1)} h_j^{(1)}\right) \end{cases} \tag{7-16}$$

其中，$f(x)$依然采用 Sigmoid 函数：$f(x)=1/(1+e^{-x})$，其对数似然函数也会随着多隐层有相应变化，通过对式(7-15)的似然函数 $\log p(v)$最大化，结合式(7-16)求得的各层节点状态，利用随机梯度上升策略求概率极大值，同时调整网络的权重参数，公式如下：

$$\begin{cases} \Delta w^{(1)} = \frac{\partial \log P(v;\varphi)}{\partial w^{(1)}} = \eta(E_{p\text{data}}[vh^{(1)^{\mathrm{T}}}] - E_{p\text{model}}[vh^{(1)^{\mathrm{T}}}]) \\ \Delta w^{(2)} = \frac{\partial \log P(v;\varphi)}{\partial w^{(2)}} = \eta(E_{p\text{data}}[h^{(1)}h^{(2)^{\mathrm{T}}}] - E_{p\text{model}}[h^{(1)}h^{(2)^{\mathrm{T}}}]) \\ \Delta w^{(3)} = \frac{\partial \log P(v;\varphi)}{\partial w^{(3)}} = \eta(E_{p\text{data}}[h^{(2)}h^{(3)^{\mathrm{T}}}] - E_{p\text{model}}[h^{(2)}h^{(3)^{\mathrm{T}}}]) \end{cases} \tag{7-17}$$

其中，$E_{p\text{data}}/E_{p\text{model}}$分别表示原始数据和网络模型分布的期望，简单情况下 $E_{p\text{data}}$由可视层输入数据通过模型第一次自底向上抽样状态来计算，而 $E_{p\text{model}}$可取网络自顶向下和自底向上的多次运行后各层模型生成状态值来计算。模型最后统一更新网络参数的迭代公式如下：

$$\Psi \leftarrow \Psi + \Delta\Psi \tag{7-18}$$

其中 $\Psi=\{w^{(1)},w^{(2)},w^{(3)}\}$，$\Delta\Psi=\{\Delta w^{(1)},\Delta w^{(2)},\Delta w^{(3)}\}$。

算法 7-3 描述了深玻尔兹曼机(DBM)的训练算法。

算法 7-3 深玻尔兹曼机训练过程

输入：训练集 $V=\{v_1,v_2,\cdots,v_m\}$，迭代次数 iter

输出：训练完成的 DBM 深网络

过程：

1：在(0,1)范围内随机初始化当前网络所有连接权 $w_{ij}^{(n)}$，$n=1,2,3$
2：**repeat**
3：**for** all $v_i \in V$ **do**
4：　　根据当前参数和式(7-16)计算自底向上和自顶向下抽样各神经元输出状态；//第一次自底向上的各隐层状态值作为数据产生，多次运行后的状态值作为模型生成
5：　　利用式(7-17)和式(7-18)的学习规则更新网络权值参数 Ψ
6：**end for**
7：**until** 网络训练达到迭代数 iter

总之，DBM 模型可以看作是一个多层无向图，每一中间层之间节点全都互有反馈，以至于 DBM 的网络结构较为复杂。在实际应用中，DBM 模型能模拟更为复杂的数据，但存在训练缓慢的问题。

7.4 栈式自动编码器

2007 年，Bengio 等人借鉴 DBN 模型的思想，构造三层自监督的 BP 网络代替 RBM 结构，连接成一种新的深网络结构，称为栈式自动编码器(Stacked Auto Encoder，SAE)。SAE 的训练过程和 DBN 相似，也包含了预训练和微调两个阶段。不同之处在于，DBN 的预训练由多个 RBM 堆叠执行，而 SAE 的结构包含了多个基本的自动编码结构(三层结构，如图 7-5(b)所示)，每一个基本自动编码结构是一个三层 BP 神经网络，包括输入层、隐层和重构层。本节首先阐述自动编码原理，然后具体描述栈式自动编码器的详细结构和算法。

7.4.1 自动编码器

通常在一些分类任务中，对于神经网络的输入样本 x 都带有标签值 $\hat{y}$，这样可以根据减小当前网络预测值 y 和标签 $\hat{y}$ 之间的均方误差来调整网络中各层参数，直到网络收敛稳定。在有监督的神经网络中，对于每个训练样本 $(x,\hat{y})$，标签 $\hat{y}$ 一般是收集到的准确输出。但是如果现在只有无标签数据，要想使用同样的思想训练网络，那么这个误差值应该怎么得到呢？自然地，可以假设有一种模型，它可以是网络输出与输入表达一致，然后直接使用这种对数据的重构 $\hat{x}$ 与原始数据 x 之差作为误差度量，再通过最小化这个重构误差来调整网络参数，这样的模型称为自动编码器(Auto Encoder，AE)。自动编码器是一种无监督学习模型，训练数据本来是没有标签的，令每个样本的标签为原始数据 x，可以理解为其标签是样本数据本身。图 7-5 对普通前馈神经网络和自动编码器原理进行了比较。

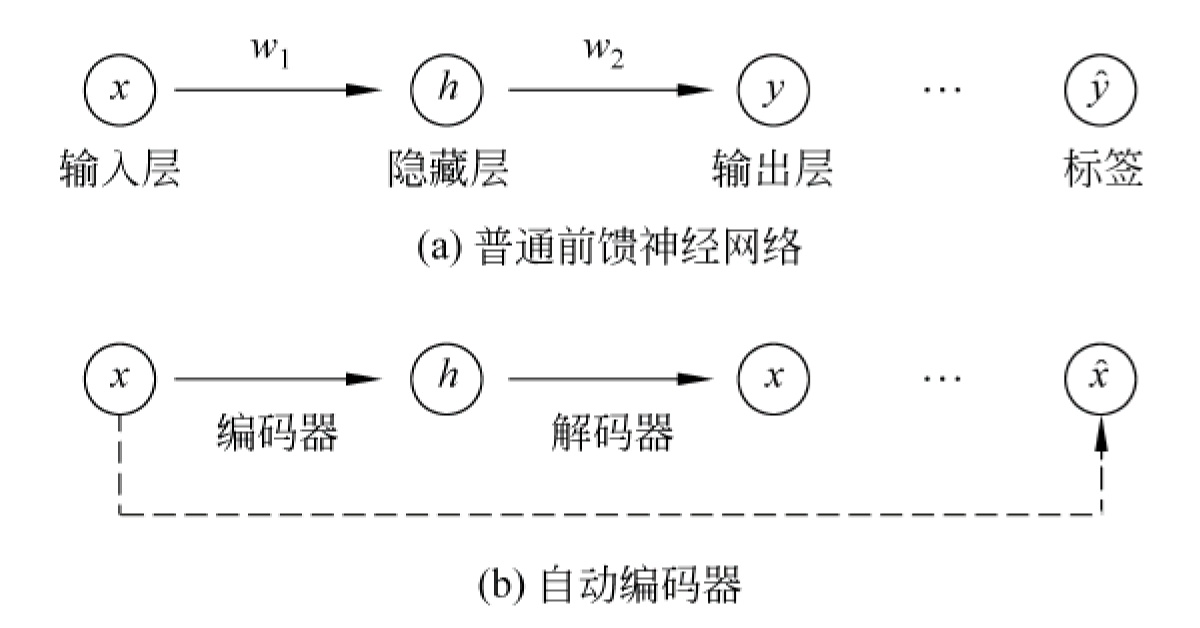


图 7-5 普通前馈神经网络和自动编码器

从图中可以看出，自动编码器就是一种尽可能重建输入信号的神经网络，原始数据通过输入到一个编码器(encoder)，就可以得到数据的隐特征表达，然后隐特征继续通过解码器(decoder)，得到对原输入数据的重构信息。它的训练包含如下过程：

(1) 输入层到隐藏层的编码过程：

$$h_j = f(w_{ij}x_i + b_j) \tag{7-19}$$

(2) 隐藏层到输出层的解码重构过程：

$$\hat{x}_i = f(v_{ij}h_j + b_i) \tag{7-20}$$

(3) 计算原始数据的重构误差损失函数：

$$E(w,b) = \frac{1}{m}\sum_{r=1}^{m}\frac{1}{2}\parallel \hat{x}^{(r)} - x^{(r)} \parallel^2 \tag{7-21}$$

(4) 结合 BP 反向传播算法，根据式(7-13)更新网络连接权值 w_{ij} 和阈值参数 b_k。

7.4.2 栈式自动编码器

栈式自动编码器(SAE)在预训练中使用无监督贪婪逐层训练三层浅网络的策略(基于 BP 算法)，通过多个三层浅网络的学习有助于获得初始优化网络参数，这使得网络首先获得一个很好的局部极小区域的初始权值，从而产生内部分布式表示高层次的抽象的输入，带来更好的逐层内在特征表达。每一层单独训练参数，然后保留这些参数。同样，为了得到更好的训练结果，预训练完成过后，SAE 结合标签，通过 BP 算法对网络所有层的参数进行有监督的全局微调，使训练结果根据目标输出得到改善。下面分析栈式自动编码器 SAE 的训练过程：

(1) 如图 7-6 所示，首先采用三层自编码网络(自监督的 BP 算法)，先训练从输入层到 $h^{(1)}$ 层的参数。

(2) 如图 7-7 所示，一个自动编码器训练完成后，$h^{(1)}$ 层的隐特征将直接作为下一个三层自动编码器的输入，接着训练从 $h^{(1)}$ 层到 $h^{(2)}$ 层的参数。

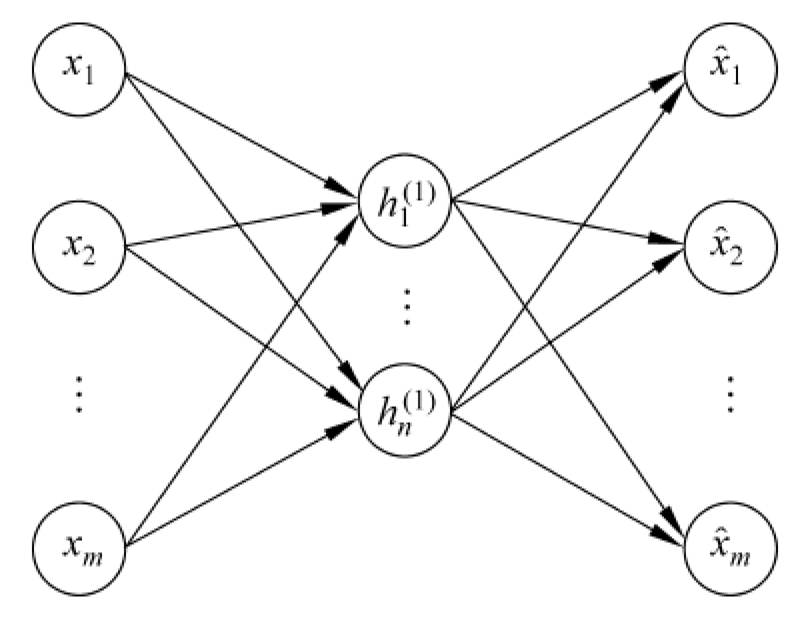


图 7-6 SAE 输入层到第一隐层的训练

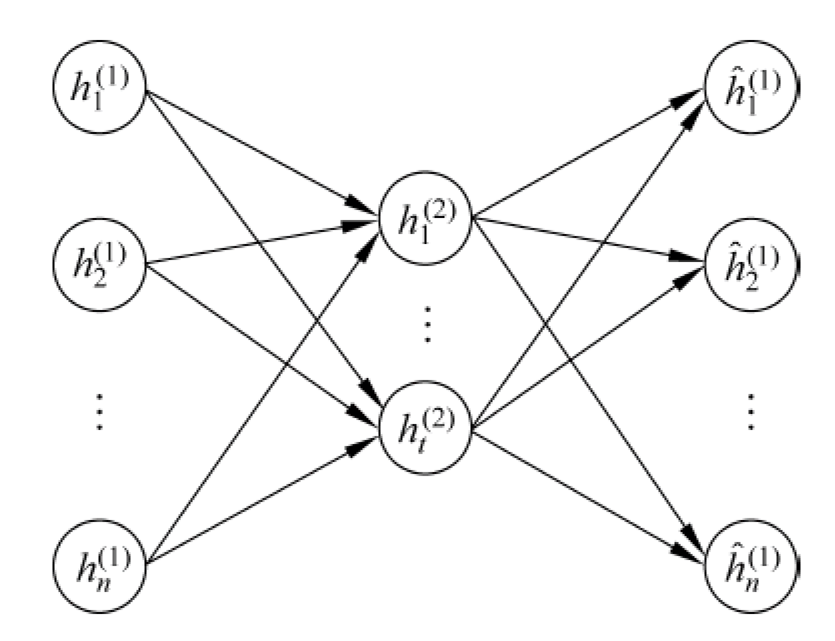


图 7-7 SAE 第一隐层到第二隐层的训练

(3) 同理，根据第(2)步，依次逐层贪婪学习，直到最后一个三层自动编码器训练完毕，则 *SAE* 的预训练结束。算法 7-4 描述了栈式自动编码器的预训练算法。

算法 7-4 栈式自动编码器预训练过程

输入： 训练集 $X=\{x_1, x_2, \cdots, x_m\}$，网络学习率 η，AE 个数 N，网络迭代次数 iter

输出： 训练完成的多个 AE 组合的深网络

步骤：

1：**for** $n = 1 \rightarrow N$ **do**

2：　　在(0,1)范围内随机初始化当前 AE 的编码权值 w_{ij}、解码权值 v_{ij} 和阈值 b_i

```
    repeat
        for all x_i ∈ X do
            根据当前参数和式(7-19)、式(7-20)计算当前 AE 的三层前向网络输出x̂_i
            根据式(7-21)计算当前 AE 输入层 x_i 和输出层x̂_i 的均方误差 E(w,b)
            根据式(7-13)更新当前 AE 的三层前向网络的权值 w_ij，v_ij 和阈值 b_i
        end for
    until 网络迭代次数达到 iter
end for
```

(4) SAE 预训练完成之后，除去网络所有的解码过程，再将网络整体连接成前向深层 BP 网络结构，利用最后一个隐层的特征数据 $h_1^{(k)},h_2^{(k)},\cdots,h_g^{(k)}$（假设网络一共 k 层，最后一个隐层有 g 个节点），结合原始数据的标签信息，接入一个分类层，实际应用中分类层常采用 SoftMax 策略，最终通过 BP 算法来训练模型执行原始数据的多分类任务，如图 7-8 所示。该过程中网络会整体微调参数，训练过程同有监督深层神经网络的学习一致。

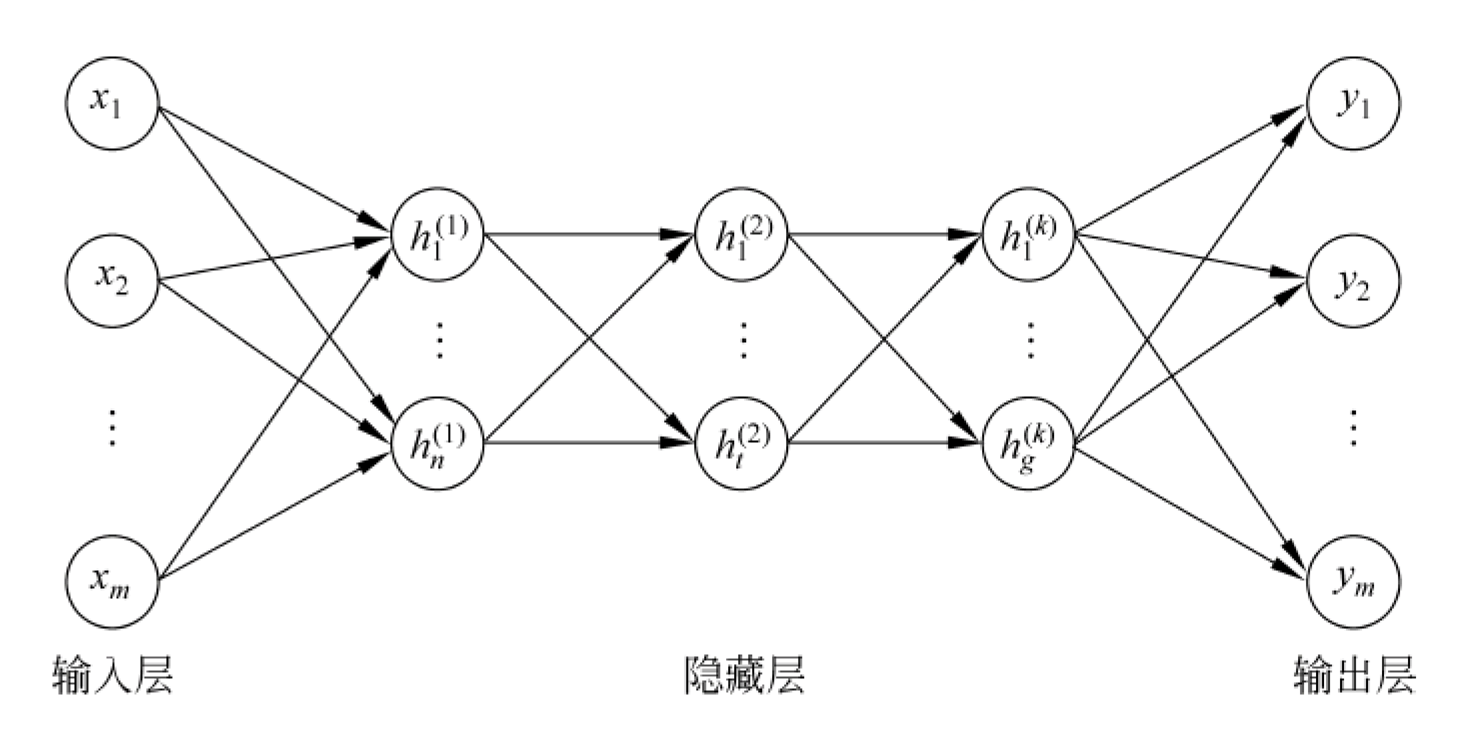


图 7-8　接入分类层的栈式自动编码器

算法 7-5 描述了栈式自动编码器(SAE)的微调学习算法。

算法 7-5　栈式自动编码器微调过程

输入：训练集 $X=\{x_1,x_2,\cdots,x_m\}$，标签集 $Y=\{y_1,y_2,\cdots,y_s\}$，网络学习率 η，每批训练样本数 batch_size，迭代次数 iter

输出：训练完成的栈式自动编码网络

步骤：

```
根据算法 7-4 对 SAE 进行预训练，并将预训练完成的网络展开为前向深层 BP 神经网络
去除网络中所有解码结构，加载预训练完成的各层 AE 权值 w_ij 和阈值 b_i
将整个训练集 X 和标签集 Y 按 batch_size 大小划分成 N 批      //随机梯度下降策略
repeat
    for n = 1 → N do
        for all X_n⊂X，Y_n⊂Y do                              //第 n 批训练集
            根据式(7-10)和式(7-11)计算当前批次样本集的网络前向输出Ŷ_n
            根据式(7-12)计算网络输出Ŷ_n 和数据标签 Y_n 的误差 E
            根据式(7-13)更新网络的连接权值 w_ij 和阈值 b_i
```

```
        end for
    end for
until 网络迭代次数达到最大迭代次数 iter 或所有输出误差小于给定误差限
```

7.5 卷积神经网络

卷积神经网络(Convolutional Neural Network,CNN)设计的灵感来源于 1962 年 Hubel 和 Wiesel 通过对猫视觉皮层细胞的研究,他们提出了感受野(Receptive Field)的概念,也称为视觉简单细胞感受野。1984 年,日本学者 Fukushima 基于简单细胞感受野概念提出了神经认知机模型,实现了第一个原始的卷积神经网络模型,也是感受野认知概念与人工神经网络结合的初次应用。卷积神经网络是为识别二维信号特别是图像而设计的一个多层感知器,这种网络结构对平移、比例缩放、倾斜或者其他形式的变换具有高度不变性,它通过一系列局部二维滤波器能够保持图像的局部空间二维关系和提取二维内在特征表示。卷积神经网络中的卷积和池化操作使得整个网络对二维图像输入的抗畸变容错能力变得更强。

卷积神经网络是一种前馈深层局部连接神经网络,通过局部的滤波与池化的交替运算多次提取内在特征,然后全连接到一个分类层,卷积神经网络模型的训练可分为两个基本过程:前向传播的卷积、池化、加权求和等操作计算输出值,以及误差反向传播调整权值和阈值。对于 CNN 的图像分类应用,其中前向传播计算输出值的过程就是:输入一张图像,对输入的图像先进行一系列交替的卷积和池化操作,实现对输入图像的特征提取,然后再将提取的特征送入到分类层来训练分类器。而误差反向传播调整权值和阈值这一过程就是:先利用整个网络输出与期望输出计算误差目标函数 E,类似 BP,误差反向逐层加权传递就得到每个权值所对应的梯度,最后用所得的梯度去更新层间连接权值和神经元阈值项,权值和偏置更新的通用公式如下:

$$\begin{cases} w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \\ \Delta w_{ij} = -\eta \dfrac{\partial E}{\partial w_{ij}} \\ b_i \leftarrow b_i + \Delta b_i \\ \Delta b_i = -\eta \dfrac{\partial E}{\partial b_i} \end{cases} \tag{7-22}$$

由于局部连接使得卷积神经网络的前向计算与误差反向传播的传统前馈神经网络略有不同。

本节首先描述矩阵的卷积和池化过程,然后分析卷及神经网络的具体结构和训练算法,最后给出一个应用案例。

7.5.1 卷积

不失一般性,假定卷积过程如图 7-9 所示且卷积核大小是 2×2,输入的特征图(feature map)X 大小是 4×4,用这个 2×2 卷积核在特征图 X 上自左向右、自上而下地滚

动卷积一遍，就可以得到一个新的大小是$(4-2+1)\times(4-2+1)=3\times3$的特征图$Y$，其中卷积核移动步幅大小是1，每次卷积都是对应元素相乘再相加(如图7-9右边所示)。新得到的特征图Y是卷积核与上一层的特征图X对应位置的神经元的输出加权求和，然后加上偏置项，再通过激活函数(activation function)变换后得到。

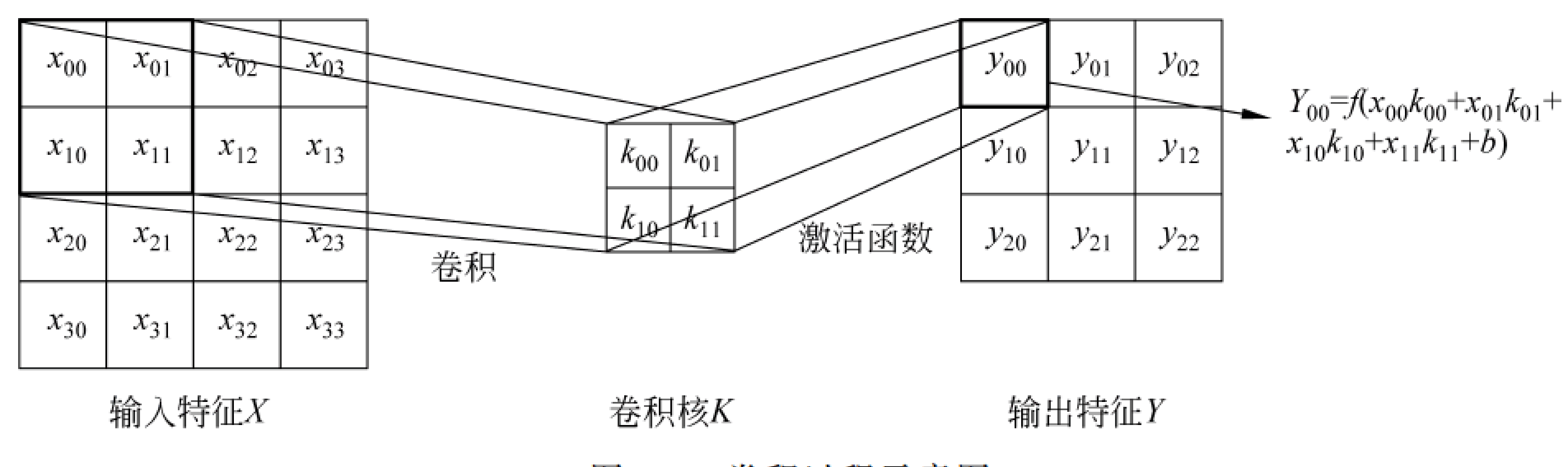


图7-9　卷积过程示意图

在卷积神经网络中，每个卷积层的特征图的大小由卷积核和上一层输入的特征图的大小决定，假如上一层的特征图大小是$n\times m$，卷积核的大小是$k_n\times k_m$，卷积核移动步幅大小是1，则当前层的特征图大小是$((n-k_n)/s+1)\times((m-k_m)/s+1)$。在卷积神经网络的实际应用中通常可以采用两种卷积方式：一种是如上所述常用的窄卷积，用conv2(X,Y,'vaild')表示；另一种就是宽卷积，在卷积运算前，先将输入的特征图X在上下各添加k_n-1行零向量，左右各添加k_m-1列零向量，保证卷积前后的特征图大小一样，然后再卷积，用conv2(X,Y,'full')表示。

在图7-9中，与输入特征图X中x_{00},x_{01},x_{10},x_{11}这4个位置的神经元相连的卷积核的权值k_{00},k_{01},k_{10},k_{11}在卷积核滚动的过程中又可以被其他的神经元使用，而这体现了卷积神经网络权值共享的特点。权重共享可以减少网络中大量的参数，从而减少存储网络时所耗费的存储容量。而且在卷积操作的过程中，下一层的特征图中的每个神经元只与上一层的特征图中卷积窗口内的神经元相连接，并不是同时与上一层特征图中所有的神经元相连接，这种上一层特征图中一个区域的神经元与下一层特征图中一个神经元直接相连的方式是强制使用局部连接模式来利用图像的空间局部二维特性，从而体现出了卷积神经网络的另一个特点——稀疏连接与空间二维保持。图7-10是邻接层稀疏连接和全连接对比图，可以看到，在上方的稀疏连接图中，神经元S_3只与X_2,X_3,X_4相连接，其他的神经元也只与前一层的部分神经元相连接；在下方的全连接图中，神经元S_3则与前一层的所有神经元连接。

在卷积操作的过程中，由于只是通过一个局部的感受野(卷积窗口)去感受外界的图像信息，网络中的神经元只感受局部的图像区域，然后在网络的更高层将这些感受不同局部区域的神经元综合起来，就可以得到全局的信息，因此，卷积神经网络具有极强的捕捉局部二维特征的能力，也具有捕捉平移变换目标的能力。

7.5.2　池化

在通过卷积获得图像的特征之后，就可以利用这些特征进行分类，从理论上来说，虽

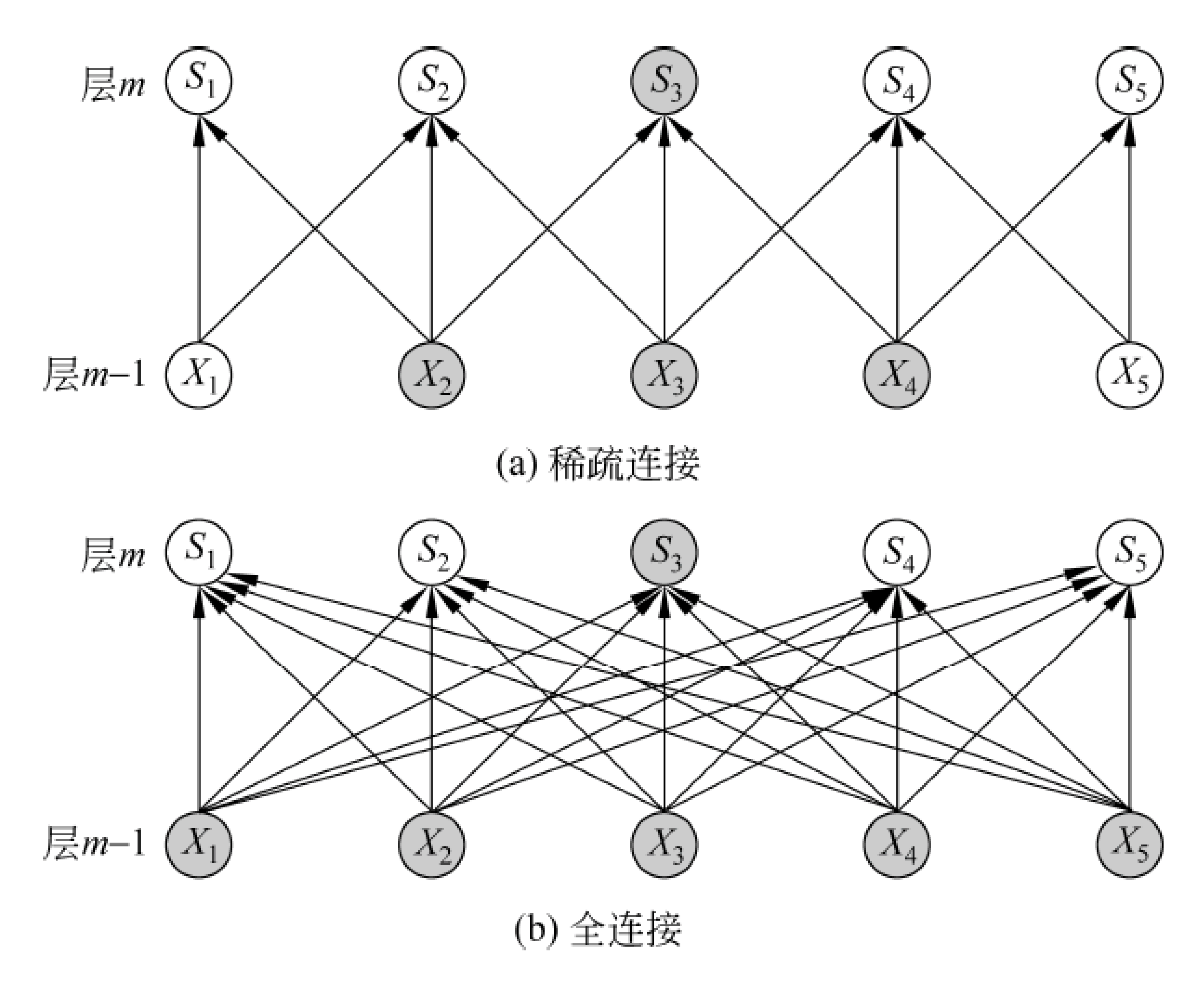


图 7-10 邻接层稀疏连接与全连接

然可以直接使用所有这些特征去训练分类器，但是由于特征图是 2D 图，如果将所有的特征全部变成一维，则卷积特征维度非常高，直接用这些特征去训练分类器，就可能会产生很大的计算量，并且也可能会产生过拟合（overfitting）的情况。为此引入了池化（pooling，也称为子采样，sub-sampling）操作，在捕捉高级特征的同时降低计算量。图像的局部重复性使得一个区域有用的特征可以用来描述另一个区域。因此，为了描述大尺度的图像，一个很自然的想法就是对不同位置的特征进行聚合统计，即池化操作。例如，对上一层特征图进行采样处理，而采样的方式就是对上一层的特征图中的不同位置相邻小区域内的特征进行聚合统计，统计的方式通常有两种，一种是求一个小区域内的特征的平均值，另一种就是求一个小区域内的特征的最大值。池化后将会得到新的特征图，此时的特征图的维度相对前一层的特征图则会低得多，池化的操作记为 $S=\mathrm{down}(X)$。

以图 7-11 中具体的池化操作为例，输入的特征图 X 大小是 4×4，池化的窗口大小是 2×2，池化窗口通常以不重叠的方式自左向右、自上而下的方式移动，此处移动步幅大小即为 2，采用平均池化的采样方法，池化窗口在特征图 X 上移动一遍，就会得到一个新的大小是 2×2 的特征图 S，而特征图 S 的大小只有上一层的特征图 X 的 1/4（注意：2×2 的池化窗口使得行列都减半）。

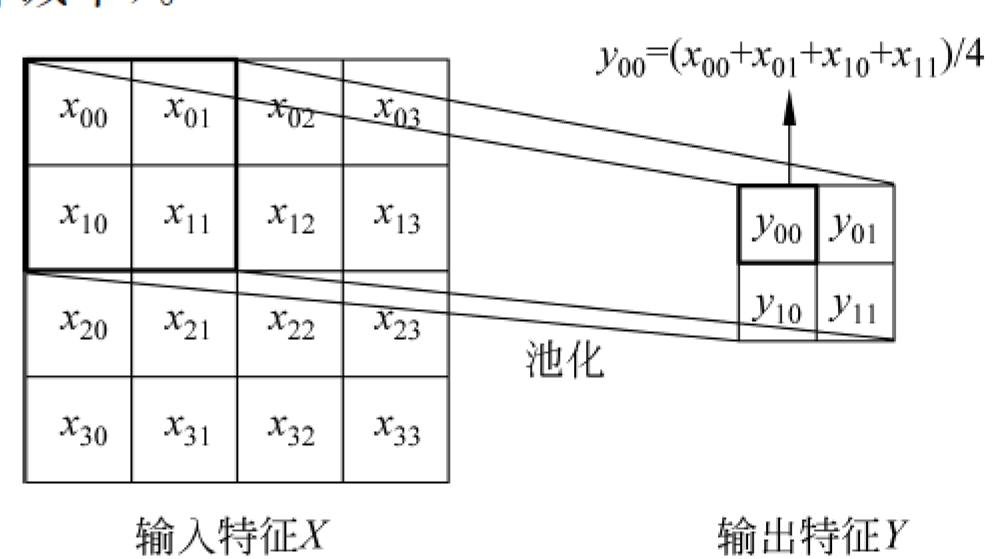


图 7-11 池化过程示意图

池化操作可以很好地降低网络中的特征维度,从而降低整个卷积神经网络模型的复杂度。除此之外,池化操作还可以起到提取图像二次高级特征的作用。

7.5.3 CNN 训练过程

由于卷积神经网络也是一种前馈局部连接神经网络,整个网络模型在训练的过程中也包括了两个过程,一个是前向传递计算输出值,另一个则是误差反向传播调整网络的权值和阈值。为了便于理解,以图 7-12 作为一个卷积神经网络的具体例子,它由 1 个输入层、2 个卷积层(卷积运算方式均为窄卷积)、2 个池化层(采用平均池化方法)、1 个全连接层以及 1 个输出层构成,下面将以这个具体的卷积神经网络结构为例来分析卷积神经网络的前向计算输出值和反向误差传播调整权值与阈值这两个过程。

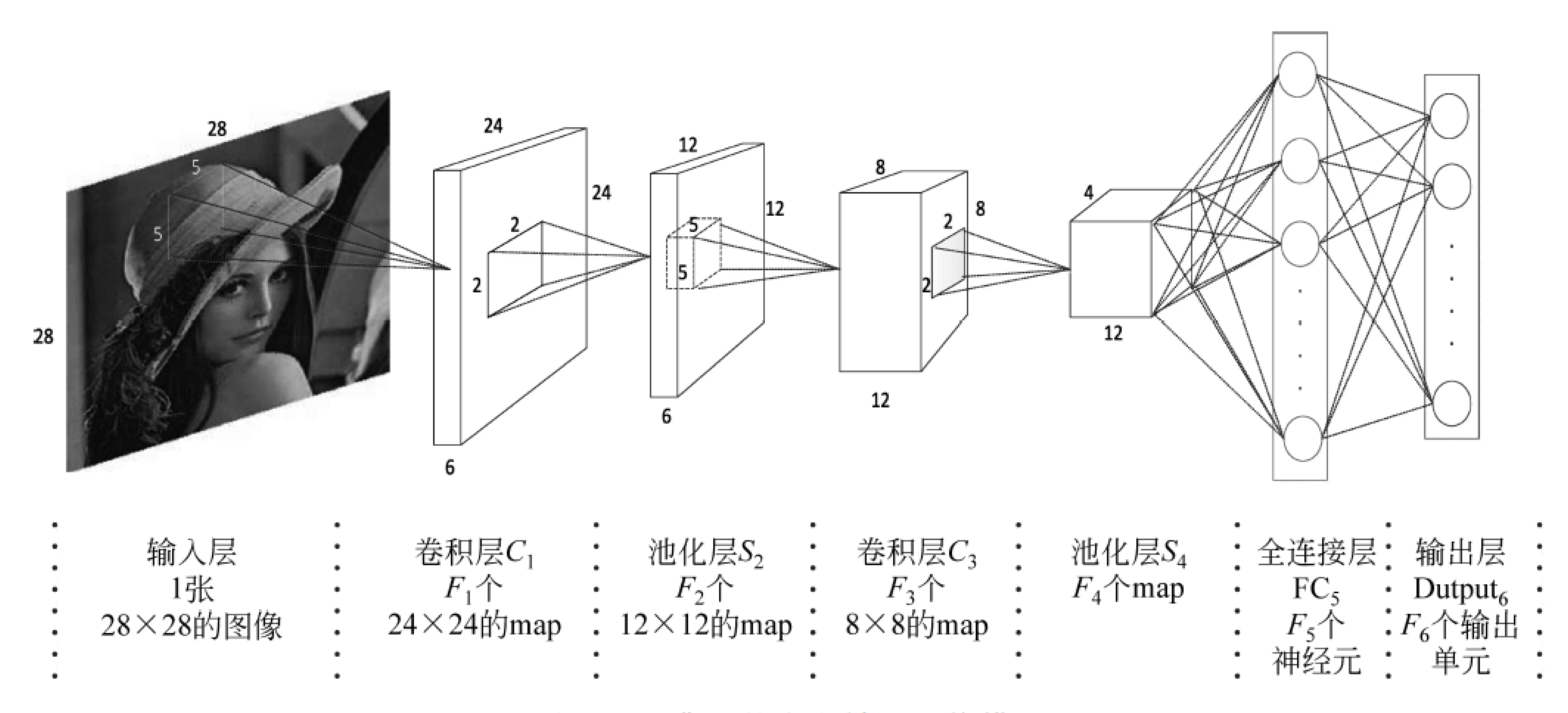


图 7-12 典型的卷积神经网络模型

1. 前向传递计算各层输出

1) C_1 层

C_1 层即第一个卷积层,图 7-12 中卷积神经网络的输入层是 28×28 的图像,经过 F_1 个大小是 5×5 的卷积核 $K_{ij}^{(1)}(i=1,2,3;j=1,2,\cdots,F_1)$卷积后,生成 F_1 个大小是 24×24 的特征图。C_1 层的每一个卷积计算为

$$C_j^{(1)}=\Big(\sum_{i\in M_j^{(1)}}X_i\times K_{ij}^{(1)}\Big)+b_j^{(1)} \tag{7-23}$$

$$a_j^{(1)}=f(C_j^{(1)}) \tag{7-24}$$

其中,$K_{ij}^{(1)}$表示第一个卷积层中第 j 个特征图与输入的第 i 个特征图之间相连的卷积核。$b_j^{(1)}$ 表示第一个卷积层第 j 个特征图的阈值项。$M_j^{(1)}$ 表示在上一层的特征图中被选中进行第 j 个卷积运算的特征图集合。本节分析的卷积神经网络中,输入层输入的假定是一幅 3 通道的 RGB 图像,将每个通道看成下一层网络的输入特征图,故 $M_j^{(1)}=\{1,2,3\}(j=1,2,\cdots,F_1)$。$f(\cdot)$表示激活函数,$a_j^{(1)}$ 表示第一个网络层第 j 个特征图。

2）S_2 层

S_2 层将 C_1 层卷积计算后的特征图进行池化。设池化窗口大小为 2×2，窗口移动步幅大小是 2，将大小是 24×24 的特征图池化成大小是 12×12 的特征图，特征图的数量和 S_2 层数量一样。S_2 层的计算输出值的方式为

$$S_j^{(2)} = \beta_j^{(2)} \mathrm{down}(a_j^{(1)}) + b_j^{(2)} \tag{7-25}$$

$$a_j^{(2)} = f(S_j^{(2)}) \tag{7-26}$$

其中，$\beta_j^{(2)}$ 和 $b_j^{(2)}$ 表示第二个网络层（池化层）第 j 个特征图的放缩因子和阈值，$f(\cdot)$ 表示一个非线性映射函数。

3）C_3 层

C_3 层将 S_2 层降采样及非线性映射后的特征图进行卷积操作。该层使用 F_3 个大小为 5×5 的卷积核 $K_{ij}^{(3)}(i=1,2,\cdots,F_2;j=1,2,\cdots,F_3)$ 对 S_2 层中的特征图做卷积运算，得到 F_3 个大小是 8×8 的特征图。C_3 层的卷积计算的输出值为

$$C_j^{(3)} = \Big(\sum_{i\in M_j^{(3)}} a_i^{(2)} \times K_{ij}^{(3)}\Big) + b_j^{(3)} \tag{7-27}$$

$$a_j^{(3)} = f(C_j^{(3)}) \tag{7-28}$$

4）S_4 层

S_4 层将 C_3 层卷积运算后的特征图池化。池化窗口大小为 2×2，窗口移动步幅大小为 2，将大小为 8×8 的特征图池化为大小为 4×4 的特征图，特征图的数量和 C_3 层数量一样。S_4 层的计算输出值的方式为

$$S_j^{(4)} = \beta_j^{(4)} \mathrm{down}(a_j^{(3)}) + b_j^{(4)} \tag{7-29}$$

$$a_j^{(4)} = f(S_j^{(4)}) \tag{7-30}$$

5）FC_5 层

FC_5 是全连接层，该层是一个与普通前馈神经网络一样的前向全连接层。在卷积和池化操作全部完成以后，由于得到的特征图仍然是二维（2D）图，因此需要将这些特征图按顺序先变成一维向量，然后再作为全连接层的输入。在 FC5 层一共有 F_5 个神经元，该层网络的权值为 $w_{ji}^{(5)}(j=1,2,\cdots,F_5;i=1,2,\cdots,4\times4\times F_4)$，$w_{ji}^{(5)}$ 表示的当前层的第 j 个神经元与上一层第 i 个神经元相连的权值。FC_5 层计算输出值的方式为

$$\mathrm{FC}_j^{(5)} = \Big(\sum_i w_{ji}^{(5)} \times x_i^{(4)}\Big) + b^{(5)} \tag{7-31}$$

$$a_j^{(5)} = f(\mathrm{FC}_j^{(5)}) \tag{7-32}$$

6）Output_6 层

Output_6 输出层的神经元的数量与要分类的数目有关系，图 7-11 中的卷积神经网络结构中类别数目是 F_6，故这一层的神经元的数量是 F_6 个，这一层网络的权值为 $w_{ji}^{(6)}(j=1,2,\cdots,F_6;i=1,2,\cdots,F_5)$。$\mathrm{Output}_6$ 层计算输出值的方式为

$$\mathrm{Output}_j^{(6)} = \Big(\sum_i w_{ji}^{(6)} \times a_i^{(5)}\Big) + b^{(6)} \tag{7-33}$$

$$o_j^{(6)} = f(\mathrm{Output}_j^{(6)}) \tag{7-34}$$

卷积神经网络的反向传播过程本质上与传统的 BP 神经网络基本一致，两种神经网络

均是使用梯度下降的方法来更新网络中的权值和阈值。但是它们之间最主要的区别在于卷积神经网络是部分连接,而传统的 BP 神经网络是全连接。因此卷积神经网络在反向关于连接权求导的过程中需要明确参数连接了哪些神经元;而全连接的传统 BP 神经网络中相邻的两层的神经元都是全部相关联的,因此反向求导时相对来说是十分简单的。下面将按着与上述正向传导计算输出值方向相反的方向传递误差来调整各层之间的相连权值和阈值。

2. 误差反向传递调整网络参数

假设当前全连接层为 l,下一个全连接层为 $l+1$,上一个全连接层为 $l-1$,则从 $l-1$ 到 l 层有

$$a_j^{(l)} = f(u_j^{(l)}) = f\left(\left(\sum_i w_{ji}^{(l)} x_i^{(l-1)}\right) + b^{(l)}\right) \tag{7-35}$$

卷积神经网络的全连接层反向调节过程和 BP 神经网络的反向调节过程是一样的,此处不再赘述。

1) 卷积层误差及卷积核的梯度

假设当前卷积层为 l,下一层为池化层 $l+1$,上一层也为池化层 $l-1$,那么从 $l-1$ 层到 l 层有

$$a_j^{(l)} = f(u_j^{(l)}) = f\left(\sum_{i \in M_j} a_i^{(l-1)} K_{ij}^{(l)} + b_j^{(l)}\right) \tag{7-36}$$

在反向传播计算当前卷积层 l 每个神经元的误差 δ 时,都需要清楚当前 l 层的输入特征图中的哪个区域块与 $l+1$ 层的哪个输出特征图中的神经元相连接,将与 $l+1$ 层相连的神经元对应的误差 δ 加起来求和,并乘以相应连接的权值,然后再乘以 l 层对应神经元的激活函数的导数,这样就得到当前 l 层每个神经元对应的误差 δ。由于卷积层后面为池化层,这就使得池化层中的每个神经元对应的误差 δ 都是与卷积层中的一块区域的神经元的输出相关联,因此卷积层的一个神经元只与池化层的一个神经元相关联,而为了更加方便地计算当前卷积层 l 的误差 δ,可以先对下一层池化层的由各个神经元对应的误差 δ 构成的误差矩阵进行上采样,使得 l 层和 $l+1$ 层的误差矩阵大小相同,然后只需要用 $l+1$ 层经过上采样误差矩阵与相应的 l 层的每个神经元激活函数求导构成的导数值矩阵逐元素相乘,再乘以相应的连接权值,就可以得到当前卷积层 l 中每个特征图对应的误差矩阵。具体公式如下:

$$\delta_j^{(l)} = \beta_j^{(l+1)}(f'(u_j^{(l)}) \circ \mathrm{up}(\delta_j^{(l+1)})) \tag{7-37}$$

其中,"$\circ$"表示逐对元素相乘(element-wise multiplication),up(·)表示上采样(upsampling)操作,上采样操作表达式如下:

$$\mathrm{up}(\boldsymbol{x}) \equiv \boldsymbol{x} \otimes 1_{n \times n}$$

假如 $\boldsymbol{x}=\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$,按池化操作区域的尺寸复制每一个元素,当池化操作是 2×2 区域均值池化,则有

$$\mathrm{up}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right) = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{bmatrix}$$

在得到当前卷积层 l 每个神经元对应的误差 δ 后，就可以计算每个偏置的梯度：

$$\frac{\partial E}{\partial b_j^{(l)}} = \sum_{u,v} (\delta_j^{(l)})_{uv} \tag{7-38}$$

其中 u、v 取值范围就是从 1 开始依次到当前层特征图的行列数。

由于卷积神经网络中很多连接的权值是共享的，因此，对于一个给定的权值，只需要求所有与该权值有联系(权值共享的连接)的神经元的梯度，然后再对这些梯度进行求和。因此每个卷积核权值的梯度计算公式如下：

$$\frac{\partial E}{\partial K_{ij}^{(l)}} = \sum_{u,v} (\delta_j^{(l)})_{uv} (p_i^{(l-1)})_{uv} \tag{7-39}$$

其中$(p_i^{(l-1)})_{uv}$表示在计算 l 层的第 j 个特征图中(u,v)位置的神经元输入值的时候，上一层第 i 个特征图 $a_i^{(l-1)}$ 在进行卷积操作时与卷积核 $K_{ij}^{(l)}$ 逐元素相乘的一个区域块。

上面的公式在 MATLAB 中仅需要使用如下卷积函数来实现即可：

$$\frac{\partial E}{\partial K_{ij}^{(l)}} = \mathrm{rot180}(\mathrm{conv2}(a_i^{(l-1)}, \mathrm{rot180}(\delta_j^{(l)}), \text{'valid'})) \tag{7-40}$$

2) 池化层误差及权值的梯度

假设当前池化层为 l，如果连接当前池化层的下一层是全连接层，则就可以直接使用 BP 神经网络的推导方法来推导当前池化层每个神经元的误差 δ。现在只讨论下一层 $l+1$ 层为卷积层，上一层 $l-1$ 层也为卷积层的情形。在这种情形下，当前池化层 l 的计算输出值方式为

$$a_j^{(l)} = f(\beta_j^{(l)} \mathrm{down}(a_i^{(l-1)}) + b_j^{(l)}) \tag{7-41}$$

其中 down(•)表示池化操作，即降采样(不重叠等区块划分后平均)。

前面在计算卷积核的梯度时，必须清楚当前 l 层的输入特征图中的哪个区域块与 $l+1$ 层的哪个输出特征图中的哪个神经元相连接。同理，在反向调节的过程中也必须找到当前 l 层的误差矩阵中哪个区域块对应下一层的误差矩阵中的哪个位置的误差，将下一层的误差反向加权传播，对应下面的计算公式就能简单计算。因此，这个过程也需要乘以输入区域与输出神经元之间的连接权值，而这个权值就是旋转后的卷积核。

$$\delta_i^{(l)} = f'(u_i^{(l)}) \circ \mathrm{conv2}(\delta_j^{(l+1)}, \mathrm{rot180}(K_{ij}^{(l+1)}), \text{'full'}) \tag{7-42}$$

上式在通过补 0 方式进行全尺寸卷积计算时，需要对卷积核进行旋转 180°，这就可以让卷积运算与误差反向传播的加权计算对应。而且，对当前 l 层进行前向卷积运算为窄卷积运算，因此在反向传播误差的过程中，则先需要对 $l+1$ 层的误差矩阵边界用 0 填充，使得 $l+1$ 层的误差矩阵的大小和 l 层的特征图的大小相同再卷积，这就是宽卷积运算。在 MATLAB 中对矩阵进行宽卷积运算的函数为 conv2(X,Y,'full')，不仿在此处采用这种表达。

在得到当前层的误差矩阵后，就可以对当前层的权值和阈值项的梯度进行计算。计算池化层 l 的阈值项的梯度和前面计算卷积层的梯度一样，只需要将误差矩阵中的误差相加即可：

$$\frac{\partial E}{\partial b_j^{(l)}} = \sum_{u,v} (\delta_j^{(l)})_{uv} \tag{7-43}$$

在正向传导的过程中，当前池化层 l 中的参数 β 还涉及原始降采样特征图的计算，因

此可以在正向传导计算的过程中直接保存那些降采样特征图，在反向传播时就不需要再重新计算一遍，降采样公式如下：

$$d_j^{(l)} = \text{down}(a_i^{(l-1)}) \tag{7-44}$$

由此可得参数 β 梯度为

$$\frac{\partial E}{\partial \beta_j^{(l)}} = \sum_{u,v} (\delta_j^{(l)} \circ d_j^{(l)})_{uv} \tag{7-45}$$

3. 卷积神经网络(CNN)的训练算法描述

卷积神经网络(CNN)的训练算法描述如下。

算法 7-6 卷积神经网络训练过程

输入： 训练集 $X=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，网络的学习率 η，每批训练样本的数量 batch_size，网络迭代次数 iter

输出： 训练完成的卷积神经网络模型(网络结构与参数)

步骤：

1：在(0,1)范围内随机初始化网络的权值和阈值
2：**repeat**
3：　将整个训练集 X 按 batch_size 大小随机划分成 N 批
4：　**for** $i = 1 \to N$ **do**
5：　　根据当前网络的参数和式(7-23)、式(7-34)正向传播计算每批样本的输出 o_i
6：　　根据式(7-10)至式(7-13)计算网络中全连接层的权值和阈值的梯度
7：　　根据式(7-38)、式(7-39)和式(7-43)至式(7-45)依次反向计算网络中卷积和池化层的权值及阈值梯度
8：　　根据式(7-22)更新整体网络中的权值和阈值
9：　**end for**
10：**until** 迭代次数达到 iter

7.5.4 CNN 网络构造的案例分析

AlexNet 是 Krizhevsky 和 Hinton 发表在 NIPS2012 会议上的一篇名为 *ImageNet Classification with Deep Convolutional Neural Networks* 的经典论文中提出的一个卷积神经网络模型。用 AlexNet 网络模型将 ImageNet ILSVRC-2012 大赛中的 150 万张图像分成 1000 类，并最终以 top-5 测试误差率 15.3%取得了当年大赛第一名的成绩，刷新了此前的图像分类纪录，从而一举奠定了深度学习在计算机视觉领域中的地位。图 7-13 是 AlexNet 网络模型的整体结构，它由卷积层、池化层、全连接层这些基本操作构成，与常见的卷积与池化操作成对出现不同，AlexNet 网络模型并不是每个卷积层的后面都会接一个池化层，为了避免由于网络模型的复杂度过高而产生的过拟合问题，还在网络中加入了 Dropout① 操作，而且由于单 GPU 显存容量的限制，整个网络是在两个 GPU 上并行

① Dropout：在训练的过程中，会以一定概率随机地停止一部分神经元工作，从而减少相邻层神经元之间的固定连接权作用，提高整个网络模型的泛化能力。

计算的。

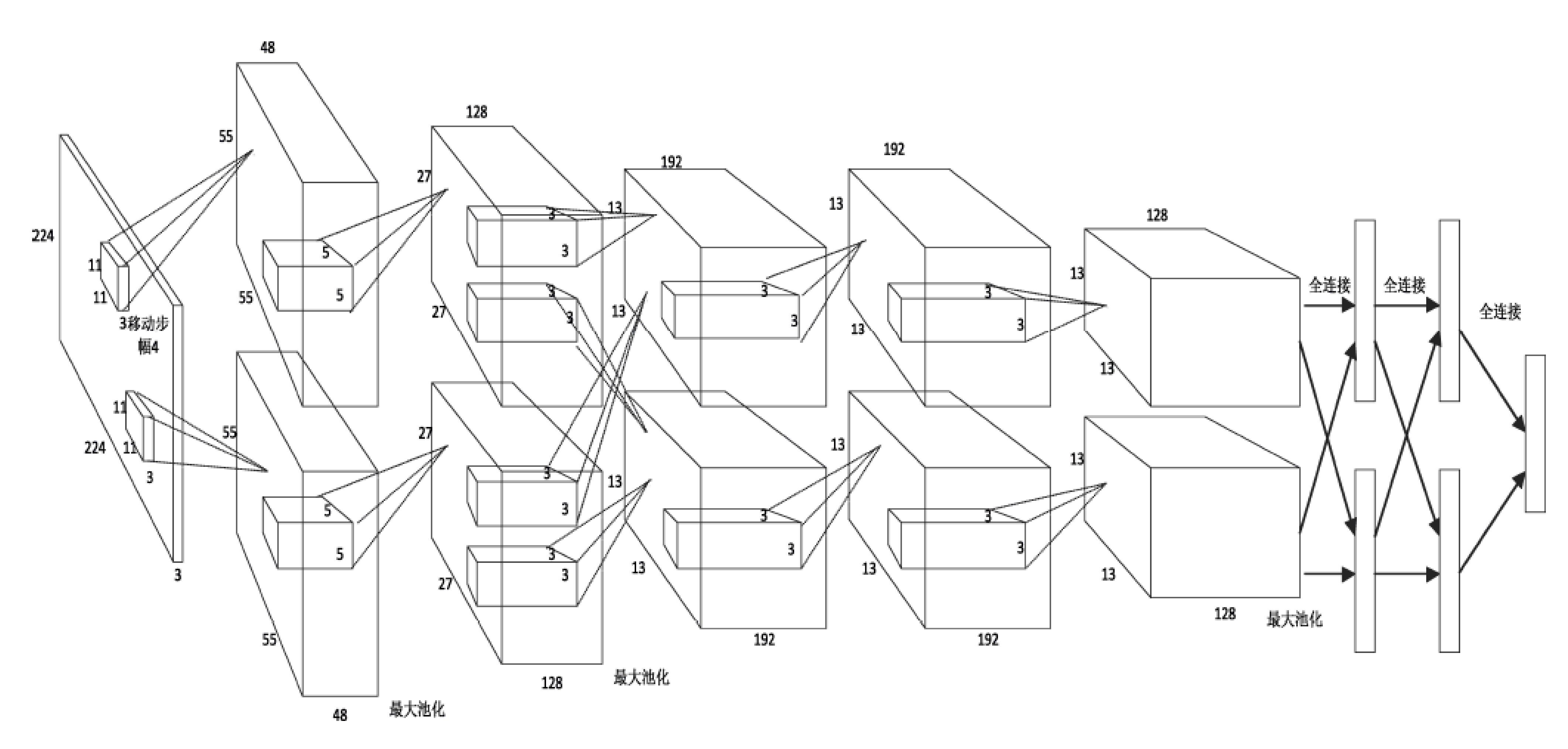


图 7-13　AlexNet 网络模型

整个 AlexNet 网络模型是一个 8 层的网络模型，其中前 5 层的卷积层和后 3 层是全连接层，最后一个全连接层的输出对应到一个具有 1000 个输出单元的 SoftMax 层用于分类。AlexNet 网络模型中的第 2、4、5 个卷积层只与驻留在相同 GPU 中的上一层的特征图相连接，在模型构造时约束第 3 个卷积层则与第 2 个卷积层中的所有特征图相连接。

整个 AlexNet 网络模型输入的是大小为 224×224 的 3 通道图像，在经过第 1 个卷积层的 96 个大小是 11×11×3 和移动步幅是 4 个像素的卷积核卷积后，得到 96 个大小是 55×55 的特征图。然后用重叠最大池化①的方法对第 1 个卷积层的 96 个特征图进行池化，得到大小是 27×27 的特征图，将这些特征图作为第 2 个卷积层的输入，然后用 256 个移动步幅是 1，大小是 5×5×48 的卷积核进行宽卷积，得到 256 个大小是 27×27 的特征图。然后再用重叠最大池化的方法对第 2 层的 256 个特征图进行池化，得到 256 个大小是 13×13 的特征图，将这些特征图作为第 3 个卷积层的输入，用 384 个移动步幅是 1，大小是 3×3×256 的卷积核进行宽卷积，得到 384 个大小是 13×13 的特征图。将这 384 个特征作为第 4 个卷积层的输入，用 384 个 3×3×192 卷积核进行宽卷积，得到 384 个大小是 13×13 的特征图，再将第 4 个卷积层的输出作为第 5 个卷积层的输入，用 256 个大小是 3×3×192 的卷积核进行宽卷积，得到 256 个大小是 13×13 的特征图。然后再对这 256 个特征图做一次重叠最大池化操作，得到 256 个大小是 6×6 的特征图。最后将经过池化得到的特征图作为全连接层的输入。

除了 2012 年的 AlexNet 网络模型，近几年还有很多以 AlexNet 网络模型为基础进行改进得到的新的网络模型，例如 Google 的 GoogLeNet 网络模型、牛津大学的 VGG 网络模型、微软研究院的 ResNet 网络模型等。这些新改进的网络相比较而言则有更深的

① 重叠最大池化：在 AlexNet 网络中，池化窗口大小是 3×3，而移动的步幅大小是 2，因此每两个相邻的池化窗口之间都有一部分重叠的像素或者神经元。

网络层次结构，对图像的识别准确率也更高。

7.6 深度学习开源框架

7.6.1 开源框架简介

随着深度学习的发展以及大数据/云计算时代的到来，一些开源的深度学习框架已被用于处理大型的复杂任务。下面简介几种目前比较主流的深度学习框架，分别是Caffe、Tensorflow、Theano、Torch、MXNet 和 DeepLearnToolBox。它们的具体情况见表 7-3。

表 7-3 几种主流深度学习框架的对比

框架	开发语言	支持接口	平台	速度	灵活性	文档	适合模型	上手难易
Caffe	C++/Python	C++，Python，MATLAB	所有系统	快	一般	全面	CNN	一般
Tensorflow	C++/Python	Python，C/C++	Linux/OSX	慢	灵活	一般	CNN/RNN	难
Theano	Python	Python	所有系统	慢	灵活	一般	CNN/RNN	容易
Torch	C，Lua	Lua/LuaJIT/C	Linux/OSX	快	灵活	全面	CNN/RNN	一般
MXNet	C++，Python，Julia，MATLAB，Go，R，Scala	C++，Python，Julia，MATLAB，JavaScript，GO，R，Scala	所有系统	快	灵活	全面	CNN	一般
DeepLearn-ToolBox	C++/MATLAB	C++/MATLAB	Linux/Windows	慢	一般	一般	CNN/DBN	容易

7.6.2 开源案例分析

在深度学习的计算机视觉应用中，经常使用到的原始数据是图像文件，图像文件格式一般是 jpg、png、tif 等，而且图像文件的大小有时也是不一样。而在 Caffe 的深度学习框架的数据输入格式通常是 lmdb、leveldb 等，此时就需要将原始的图像文件转化成适合 Caffe 输入的数据格式。现具体以将图像文件转换成 lmdb 格式的数据为例。

本例中所用的图像数据集是 cifar-10，共有 60 000 张图像，其中训练集图像有 50 000 张，测试集图像有 10 000 张，整个图像集中所有的图像可以分成 10 个类别。下面将 cifar-10 数据集中这 60 000 张图像文件转换成 lmdb 文件。

通过下面的命令，进入安装好的 caffe 目录下的 cifar10 文件夹，然后在该文件夹中编写网络的配置文件和相应的脚本文件。

```
cd caffe/examples/cifar10/
```

然后编写一个脚本程序 create_filelist. sh 用来生成训练集图像和测试集图像的清单

文件。

程序清单 7-1 生成训练集图像和测试集图像清单文件

```
#!/usr/bin/env sh
DATA=data/cifar10
TARGET=examples/cifar10
echo "Create train.txt ..."
rm -rf $TARGET/train.txt
for ((i=0;i<10;++i));
    do
        find $DATA/cifar10_train_images -name $i*.jpg | cut -d '/' \
        -f3-5 | sed "s/$/ $i/">>$TARGET/train.txt
    done
echo "Create test.txt ..."
rm -rf $TARGET/test.txt
for ((i=0;i<10;++i));
    do
        find $DATA/cifar10_test_images -name $i*.jpg | cut -d '/' \
        -f3-5 | sed "s/$/ $i/">>$TARGET/test.txt
    done
echo "All done ..."
```

然后切换到 Caffe 目录下，运行此脚本文件。最后得到的图像清单文件如图 7-14 所示。

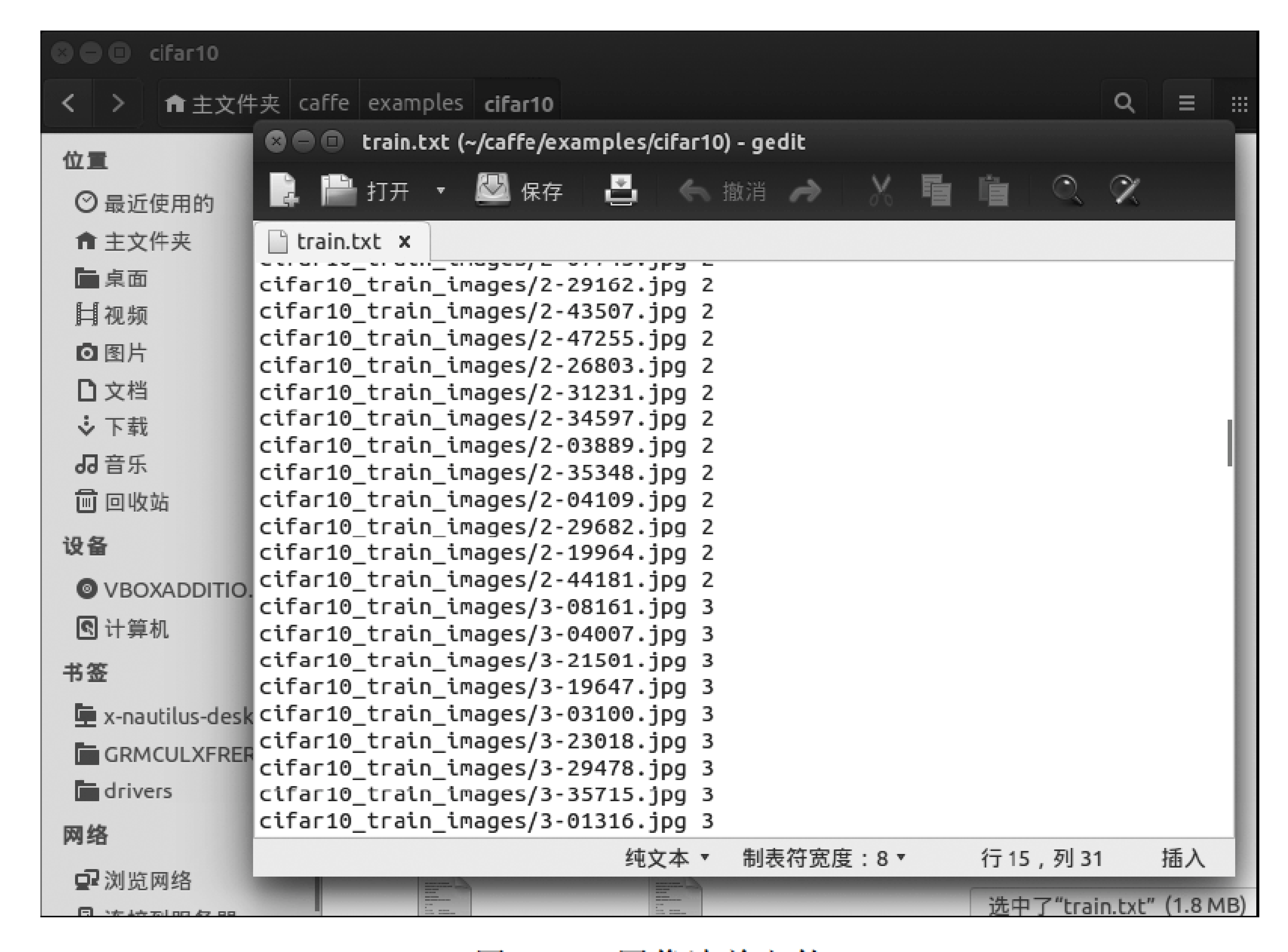


图 7-14 图像清单文件

接着再在 cifar10 文件夹中编写一个 create_cifar10_lmdb. sh 脚本程序，通过调用 convert_imageset 命令将原始图像文件数据转换成 lmdb 格式的数据。编写完成后，用和运行 create_filelist. sh 文件一样的方法运行 create_cifar10_lmdb. sh，得到 lmdb 格式的数据。运行 create_cifar10_lmdb. sh 成功后，在 cifar10 文件下生成两个文件夹，如图 7-15 所示。

cifar10_img_test_lmdb　　cifar10_img_train_lmdb

图 7-15　生成的两个装有训练数据和测试数据的文件夹

create_cifar10_lmdb. sh 脚本文件如下：

程序清单 7-2　转换数据格式

```
#!/usr/bin/env sh
TARGET=examples/cifar10
echo "Create train lmdb ..."
rm -rf $TARGET/cifar10_img_train_lmdb
build/tools/convert_imageset \
--shuffle \
--resize_height=32 \
--resize_width=32 \
/home/shihuai02/caffe/data/cifar10/ \
$TARGET/train.txt \
$TARGET/cifar10_img_train_lmdb
echo "Create test lmdb..."
rm -rf $TARGET/cifar10_img_test_lmdb
build/tools/convert_imageset \
--shuffle \
--resize_height=32 \
--resize_width=32 \
/home/shihuai02/caffe/data/cifar10/ \
$TARGET/test.txt \
$TARGET/cifar10_img_test_lmdb
echo "All done..."
```

在得到适合 Caffe 输入的 lmdb 格式的数据后，为了提高训练的速度和准确度，需要对图像数据进行预处理，这里只进行了减均值操作。其中均值是通过训练集图像得到，在 Caffe 中，提供了一个计算均值的文件 compute_image_mean. cpp，因此只需要直接使用即可：

```
sudo build/tools/compute_image_mean
examples/cifar10/cifar10_img_train_lmdb les/cifar10/mean.binaryproto
```

在得到网络模型的输入数据后，接下来开始创建模型并编写相应的配置文件。在 cifar10 文件夹中创建一个 cifar10_train. prototxt 文件，然后在该文件中编写自己的网络模型。

对于每个网络模型，先需要给网络取一个名字：

```
name: "CIFAR10_TRAIN"
```

接下来，进行网络模型训练的时候需要把 cifar10 的图像数据从 lmdb 格式的文件中读取出来，因此首先需要定义网络的数据层，以定义数据的读入方式：

```
layer {
    name: "cifar"
    type: "Data"
    top: "data"
    top: "label"
    include {
        phase: TRAIN
    }
    transform_param {
        mean_file: "examples/cifar10/mean.binaryproto"
    }
    data_param {
        source: "examples/cifar10/cifar10_img_train_lmdb"
        batch_size: 100
        backend: LMDB
    }
}
```

网络模型中，卷积层的编写格式如下：

```
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "data"
    top: "conv1"
    param {
        lr_mult: 1
    }
    param {
        lr_mult: 2
    }
    convolution_param {
        num_output: 32
        pad: 2
        kernel_size: 5
        stride: 1
        weight_filler {
            type: "gaussian"
            std: 0.0001
```

```
        }
        bias_filler {
            type: "constant"
        }
    }
}
```

池化层的编写格式如下：

```
layer {
    name: "pool1"
    type: "Pooling"
    bottom: "conv1"
    top: "pool1"
    pooling_param {
        pool: MAX
        kernel_size: 3
        stride: 2
    }
}
```

全连接层的编写格式如下：

```
layer {
    name: "ip1"
    type: "InnerProduct"
    bottom: "pool3"
    top: "ip1"
    param {
        lr_mult: 1
    }
    param {
        lr_mult: 2
    }
    inner_product_param {
        num_output: 64
        weight_filler {
            type: "gaussian"
            std: 0.1
        }
        bias_filler {
            type: "constant"
        }
    }
```

```
}
```

在网络中,ReLU 激活函数编写格式如下:

```
layer {
    name: "relu1"
    type: "ReLU"
    bottom: "pool1"
    top: "pool1"
}
```

由于对 cifar10 图像集分类是一个多分类问题,因此在网络最后一层使用的是 softmax_loss 层:

```
layer {
    name: "loss"
    type: "SoftmaxWithLoss"
    bottom: "ip2"
    bottom: "label"
    top: "loss"
}
```

在编写完网络模型文件后,还需要编写一个网络模型的配置文件:

```
cifar10_solver.prototxt:
net: "examples/cifar10/cifar10_train.prototxt"
test_iter: 100
test_interval: 500
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
lr_policy: "inv"
gamma: 0.0001
power: 0.75
display: 100
max_iter: 10000
snapshot: 5000
snapshot_prefix: "examples/cifar10/cifar10"
solver_mode: CPU
```

在训练网络模型所需要的训练数据、网络模型的定义文件、网络模型的配置文件全部准备齐全后,就可以正式开始训练网络模型。同样,先需要切换到 Caffe 的主目录下,然后通过下面的命令开始训练模型。图 7-16 是网络开始成功训练时显示的信息。

```
./build/tools/caffe train --solver=examples/cifar10/cifar10_solver.prototxt
```

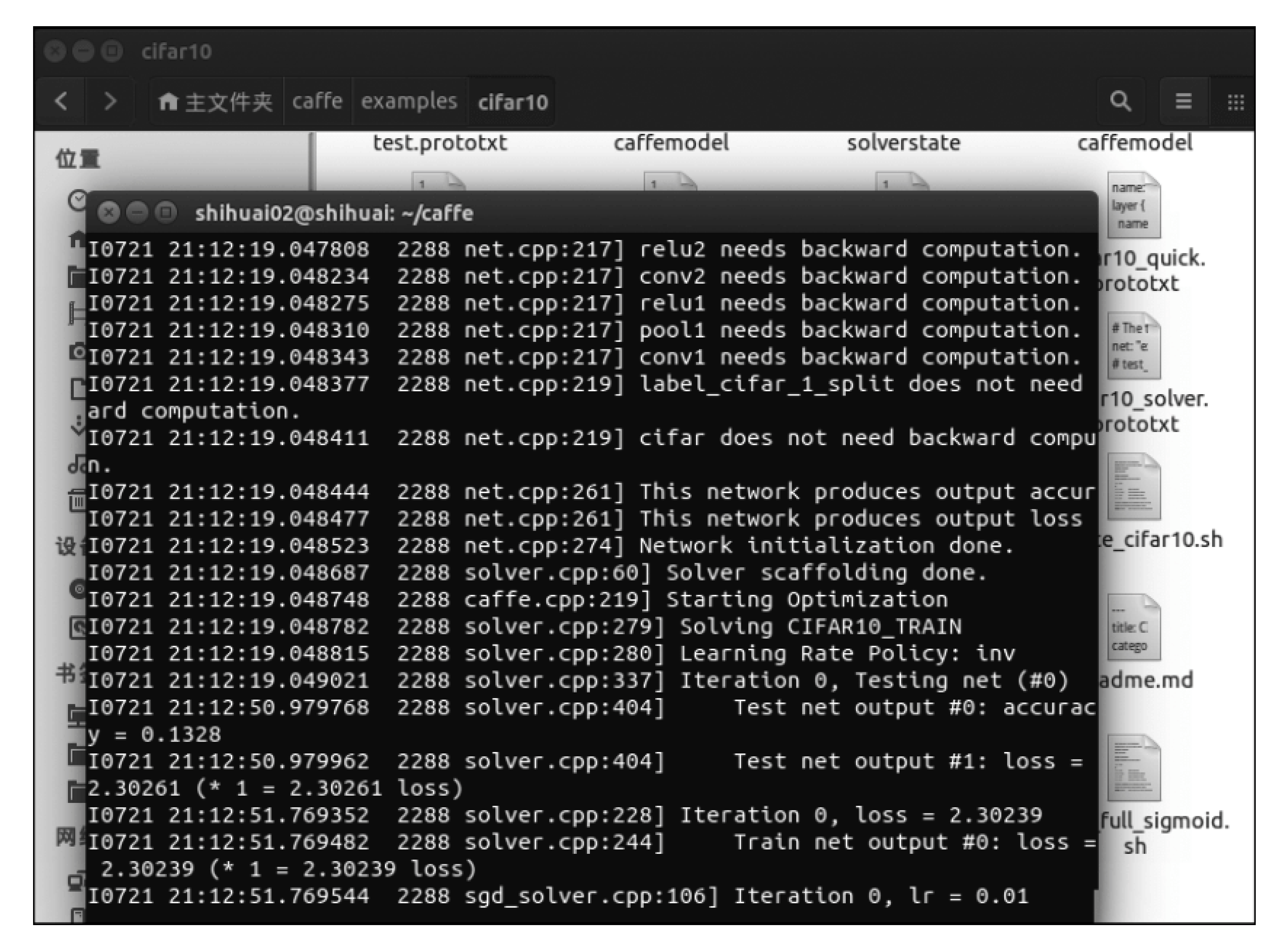


图 7-16　网络开始成功训练时显示的信息

7.7　深度学习应用技巧

以下是关于深度学习的一些技巧。

(1) 多类不平衡问题。为了避免学习结果更加趋向某一类样本,各个类别的训练样本尽量保持平衡。不平衡类训练集可以采用划分、抽样等方式将不平衡类训练集变换为平衡类训练集。

(2) 样本标签设置问题。对于神经网络来说,其激活函数都会限定神经元输出值范围为(0,1),所以一般有监督学习的情况下,对样本的标签设定值一般也要限定范围为(0,1),模型工作时使用 SoftMax 概率值来识别输出的类别。

(3) 学习率对梯度下降法的影响问题。神经网络学习率的设定一定要合适,不可太大,也不可太小。如果学习率过大,则每次迭代就有可能出现大幅度的振荡现象,也可能会在极值点两侧出现发散,误差函数值容易引起振荡;如果学习率太小,误差函数下降得很慢,导致收敛速度过慢。所以,对于网络的学习率一般都会选择(0,1)范围内的一个合适值,比如 0.0001、0.001、0.01 和 0.1 等,具体需要根据具体实验中的情况来选择一个合适的值。

(4) 大规模训练样本的随机打乱问题。大数据集的深度学习时一般采用分批的方式进行,往往没有打乱训练样本的时候,模型学习的过程中,后面的样本学习逐渐覆盖前面样本学习到的网络权值,使得它的学习结果更趋向于最后学习的类别。虽然通过大量训练迭代轮数可以相对缓和,但是也不如随机打乱样本后训练更稳定。

(5) 网络层次结构和迭代次数的设计问题。多少层网络算深度学习？多少次迭代才算合理？当前多数分类、回归等学习方法为浅层结构算法，其局限性在于有限样本和计算单元情况下对复杂函数的表示能力有限，针对复杂分类问题其泛化能力受到一定制约。深度学习必须依照实际情况控制网络结构和训练迭代次数，多方面控制模型参数，尤其是在大规模数据的任务处理中更要如此。

(6) 训练过程中目标函数梯度问题。通常采用的随机梯度下降(SGD)策略是由于所有训练样本的梯度下降法收敛速度慢，通过随机选择训练集的一个子集，计算该子集的网络输出与真实输出的平均误差梯度并调整相应的权值。可以看作对每个单独的训练子集定义不同的误差函数。标准梯度下降策略是计算所有训练样本总误差的梯度来更新权值，而随机梯度下降的权值是通过考查训练集的某个子集来更新的网络权值；标准梯度下降中权值更新的每一步对所有样本计算误差，比随机梯度需要更多的计算量；标准误差曲面有多个局部极小值，随机梯度下降可能避免陷入这些局部极小值和提高泛化能力。

(7) Dropout 技巧的使用问题。为了减少神经网络的过拟合，通常还可以在网络中加入 Dropout 层。在网络训练的时候，Dropout 以一定的概率随机地抑制一些神经元的输出，使得网络中的神经元在每次训练的时候并不是总是依赖于上一层固定的几个神经元的输出，从而使整个网络模型的泛化能力大为增强。

7.8 小　结

本章首先简单分析了深度学习的发展，其次针对几种典型的深度学习模型展开了详细介绍。再次，介绍了近年来比较主流的几种深度学习开源框架并给出了案例分析。最后，在深度学习的应用方面也给出了几点建议。下面对几种模型进行小结：

深信网(DBN)是层层堆叠多个 RBM 组成的深网络，训练过程分为两个阶段，首先通过 RBM 预训练，从输入层开始逐层为每组找到其局部最好参数的初始设置，然后再基于这些预训练的 RBM 连接起来构建整体深网络并采用随机梯度进行全局微调。

深玻尔兹曼机(DBM)借鉴了能量模型 RBM 的基本思想，它是一种通过增加 RBM 的隐层数量而构造的深网络，能够通过 Gibbs 抽样模拟复杂数据的内在特征，学习过程同 RBM 相似，但是网络层次增多，计算量较大。

栈式自动编码器(SAE)通过组合多个三层“自动编码”网络而构造深网络，首先逐层贪婪预训练，每一个三层网络的自监督学习过程同 BP 算法一致。预训练完成后对整个网络进行双向展开，再运用 BP 算法进行整体微调。

卷积神经网络(CNN)是针对二维数据而设计的一种模拟“局部感受野”的局部连接的神经网络结构，引入卷积运算实现局部连接和共享权值的特征提取，引入池化操作实现低功耗计算和高级特征提取，网络构造通过多次卷积和池化过程来形成深网络，网络的训练含有“权共享”和“稀疏”特点，学习过程类似于 BP 算法。

7.9 习　　题

1．下载 MNIST 手写体数据集，利用深度学习开源代码，自己分别设计一个深信网、栈式自动编码器和卷积神经网络，并分别进行实验测试与对比。

2．试推导 RBM 的参数更新公式(7-5)。

3．试推导卷积神经网络的卷积层和池化层的参数更新公式。

4．设计针对视频数据的卷积神经网络。

5．实验深度学习的 GPU 环境下的平台搭建。

6．实验对比深度学习的各种训练技巧对深网络性能的影响。

7．深度学习如何解决大数据分析与处理问题?

8．研究 Denoising Autoencoder 模型。

9．研究卷积深信网。

10．研究时序数据的深度学习训练与特征提取。

7.10 参考文献

[1] Kohonen T. An Introduction to Neural Computing. Neural Networks，1988，1(1)：3-16.

[2] Mcculloch W S，Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biology，1943，52(4)：99-115.

[3] 周志华. 机器学习. 北京：清华大学出版社，2016.

[4] Ackley D H，Hinton G E，Sejnowski T J. A Learning Algorithm for Boltzmann Machines. Cognitive Science，1985，9(1)：147-169.

[5] Hinton G E. A Practical Guide to Training Restricted Boltzmann Machines. Momentum，2010，9(1)：599-619.

[6] Hinton G E. Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation，2002，14(8)：1771-800.

[7] Hinton G E，Osindero S，Welling M，et al. Unsupervised Discovery of Nonlinear Structure Using Contrastive Backpropagation. Cognitive Science，2006，30(4)：725-31.

[8] Hinton G E，Salakhutdinov R R. Reducing the Dimensionality of Data with Neural Networks. Science，2006，313(5786)：504-507.

[9] Hinton G E，Salakhutdinov R R，et al. Supporting Online Material for Reducing the Dimensionality of Data with Neural Networks. Science，2006，504(5786)：504-507.

[10] Hinton G E，Osindero S，Teh Y W. A Fast Learning Algorithm for Deep Belief Nets. Neural Computation，2006，18(7)：1527-1554.

[11] Kang S，Qian X，Meng H. Multi-Distribution Deep Belief Network for Speech Synthesis[C]// 2013：8012-8016.

[12] Salakhutdinov R，Larochelle H. Efficient Learning of Deep Boltzmann Machines. 2010，9(8)：693-700.

[13] Salakhutdinov R，Tenenbaum J B，Torralba A. Learning with Hierarchical-Deep Models. IEEE

Transactions on Pattern Analysis & Machine Intelligence, 2013, 35(8): 1958-1971.

[14] Schölkopf B, Platt J, Hofmann T. Greedy Layer-wise Training of Deep Networks. Advances in Neural Information Processing Systems, 2007: 153-160.

[15] Qi Y, Wang Y, Zheng X, Wu Z. Robust Feature Learning by Stacked Autoencoder with Maximum Correntropy Criterion. ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing, 2014: 6716-6720.

[16] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. Computer Science, 2014.

[17] Szegedy C, Liu W, Jia Y, et al. Going Deeper with Convolutions[C]// Computer Vision and Pattern Recognition. IEEE, 2014: 1-9.

[18] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. Computer Science, 2015.

[19] Khalil-Hani M, Sung L S. A Convolutional Neural Network Approach for Face Verification[C]// International Conference on High Performance Computing & Simulation. 2014: 707-714.

[20] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based Learning Applied to Document Recognition. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[21] Krizhevsky A, Sutskever I, Hinton G E. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems, 2012, 25(2): 2012.

[22] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. Computer Science, 2012, 3(4): 212-223.

[23] Lecun Y, Bengio Y, Hinton G E. Deep Learning. Nature, 2015, 521(7553): 436-44.

[24] Bengio Y. Learning Deep Architectures for AI. Foundations & Trends in Machine Learning, 2009, 2(1): 1-127.

[25] Salakhutdinov R, Hinton G E. Replicated Softmax: an Undirected Topic Model. [C]// Advances in Neural Information Processing Systems 22: Conference on Neural Information Processing Systems 2009. Proceedings of A Meeting Held 7-10 December 2009, Vancouver, British Columbia, Canada. 2009: 1607-1614.

[26] Larochelle H, Bengio Y, Louradour J, et al. Exploring Strategies for Training Deep Neural Networks. Journal of Machine Learning Research, 2009, 10(10): 1-40.

[27] Taylor G W, Hinton G E, Roweis S T. Two Distributed-State Models for Generating High-Dimensional Time Series. Journal of Machine Learning Research, 2011, 12(2): 1025-1068.

[28] 余凯，贾磊，陈雨强，等. 深度学习的昨天、今天和明天. 计算机研究与发展，2013，50(09);1799-1804.

[29] Schulz H, Behnke S. Deep Learning. Kunstliche Intelligenz, 2012, 26(4): 357-363.

[30] Deng L, Yu D. Deep Learning: Methods and Applications. Foundations & Trends in Signal Processing, 2013, 7(3): 197-387.



R 语　言

R 语言是用于统计分析、绘图的语言和操作环境，它是属于 GNU 系统的一个自由、免费、源代码开放的软件。编译和运行在各种各样的 UNIX、Linux、Windows 和 MacOS 平台。从本质上讲，R 语言是 S 语言的一种实现。S 语言是由 AT&T 贝尔实验室开发的一种用来进行数据探索、统计分析和绘图的解释型语言。R 语言最初由新西兰奥克兰大学的 Robert Gentleman 和 Ross Ihaka 开发，现在由"R 核心团队"负责开发。截至 2016 年 6 月，R 语言最新的版本为 3.3.1。它包含一套连贯完整的数据分析处理算法、统计分析计算和统计制图可视化软件工具。

本章从 R 语言的下载安装(8.1 节)开始，然后学习 R 语言的基本技术，包括最常用的运行 R 语言、R 语言常用操作、包的使用(8.2 节)，R 语言的常用数据结构(8.3 节)，R 语言的编程结构(8.4 节)，以及 R 语言的数据挖掘和图形绘制包介绍(8.5 节)，最后给出一个使用 R 语言完成从数据准备预处理到数据分析的具体案例，使读者能够熟悉使用 R 语言做数据分析的整个过程。

8.1　下载和安装 R 语言

本节给出 R 语言的下载方法及其安装过程，为读者学习 R 语言建立环境。首先描述 R 语言的下载，然后描述 R 语言的安装。

8.1.1　下载 R 语言

在 R 语言的官方网站上可以下载 R 语言的源代码和 UNIX、Linux、Windows、MacOS 平台的编译好的二进制版本，下载地址是 http://www.r-project.org/的 CRAN (Comprehensive R Arichive Network)栏目下。首先选择在 China 的镜像服务器，以获取最快的下载速度，选择 Download R for Windows，再选择 base。本书使用的 R 语言版本为 3.3.1，文件名为 R-3.3.1-win.exe，根据自己的计算机操作系统体系结构的不同，安装时可以选择 32 位或者 64 位版本，大小约 70MB，如图 8-1 所示。

8.1.2　安装 R 语言

运行下载的 R-3.3.1-win.exe，在弹出的"安全警告"对话框中，单击"运行"按钮，如图 8-2 所示。

R-3.3.1 for Windows (32/64 bit)

Download R 3.3.1 for Windows (70 megabytes, 32/64 bit)

图 8-1　R 语言的 Windows 安装程序

弹出“选择语言”对话框，使用默认的“中文(简体)”，单击“确定”按钮，如图 8-3 所示。

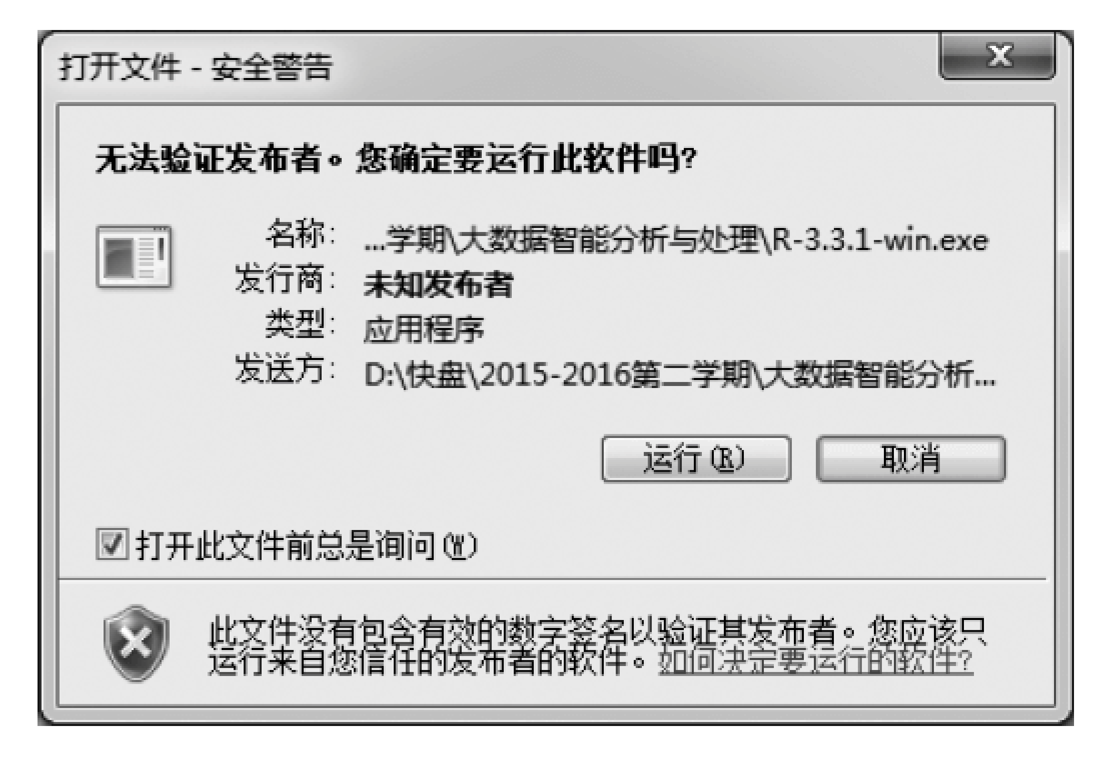


图 8-2　安全警告对话框

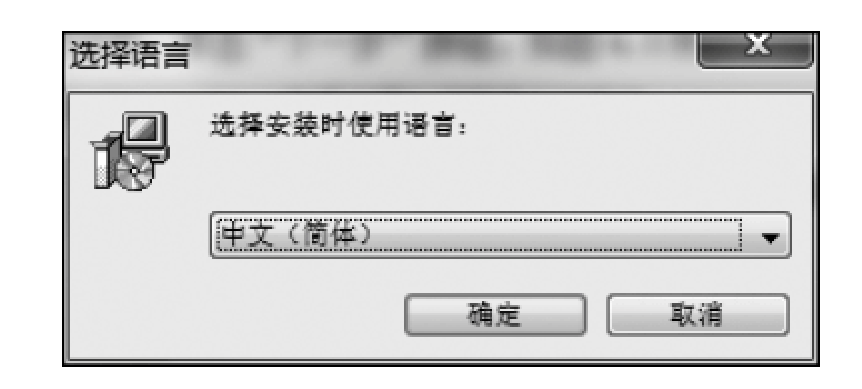


图 8-3　“选择语言”对话框

在“安装向导”的欢迎界面中，单击“下一步”按钮，如图 8-4 所示。



图 8-4　“安装向导”界面

在安装向导的“信息”界面中，阅读 GNU 通用公共许可证，单击“下一步”按钮，如图 8-5 所示。

因 R 语言整体系统较小，使用默认目录 C:\Program Files\R\R-3.3.1 进行安装，单击“下一步”按钮，如图 8-6 所示。

在“选择组件”界面中，根据计算机操作系统体系结构，本书中选择“64-bit 用户安装”，单击“下一步”按钮，如图 8-7 所示。

在“启动选项”界面中，采用默认选项 No，单击“下一步”按钮，如图 8-8 所示。

在“选择开始菜单文件夹”界面中采用默认的 R，单击“下一步”按钮，如图 8-9 所示。

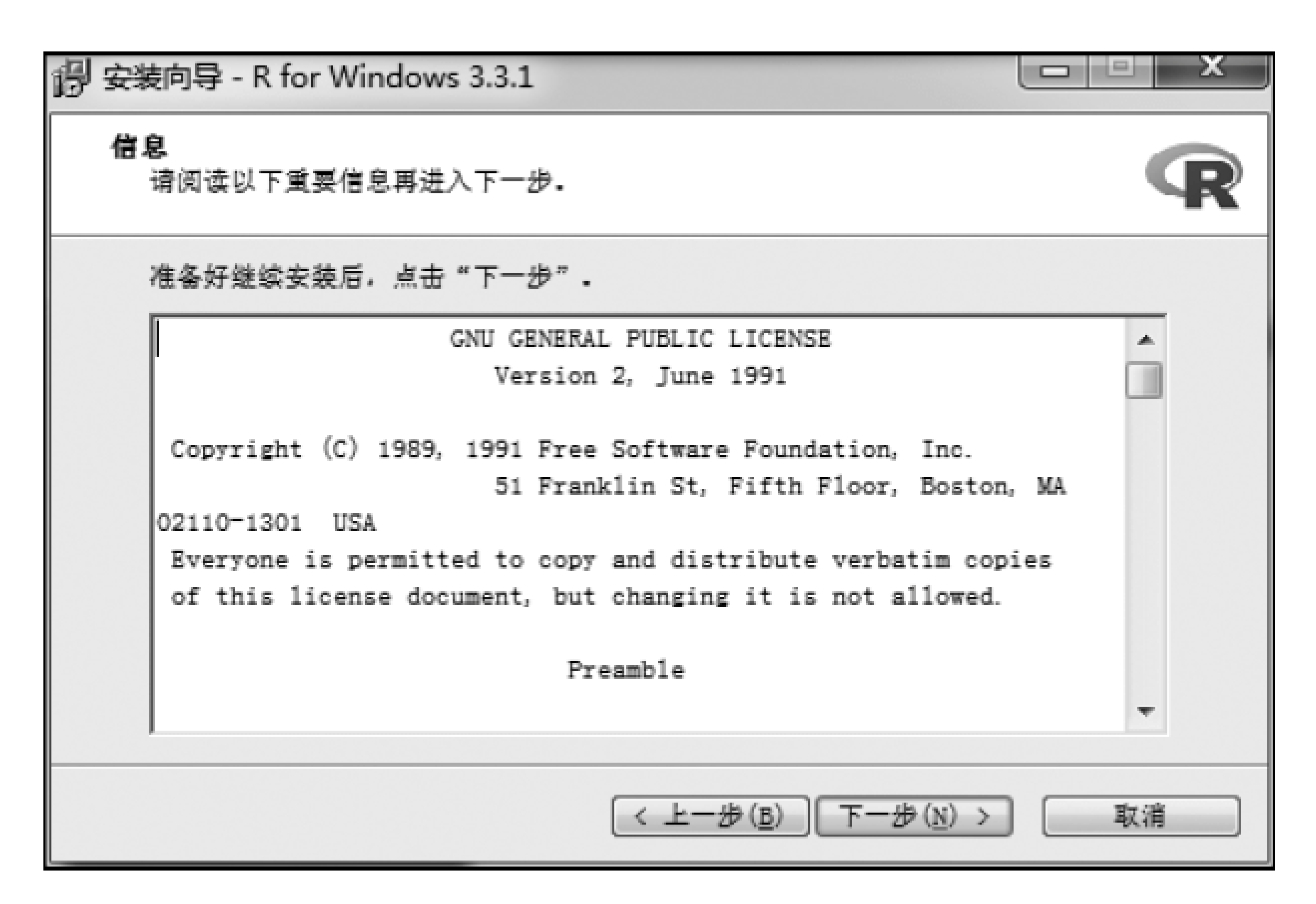


图 8-5　GNU 通用公共许可证



图 8-6　选择安装位置



图 8-7　“选择组件”界面

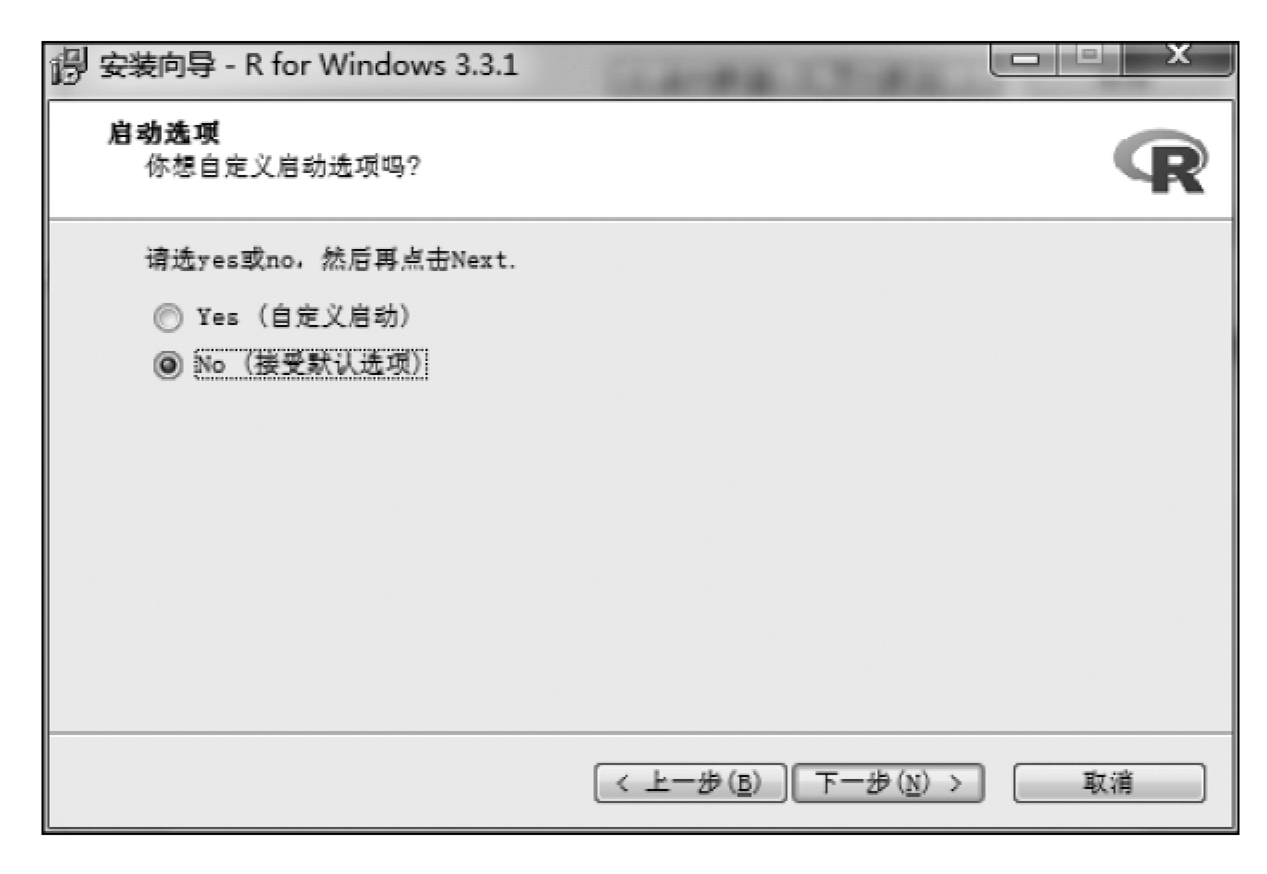


图 8-8 “启动选项”界面

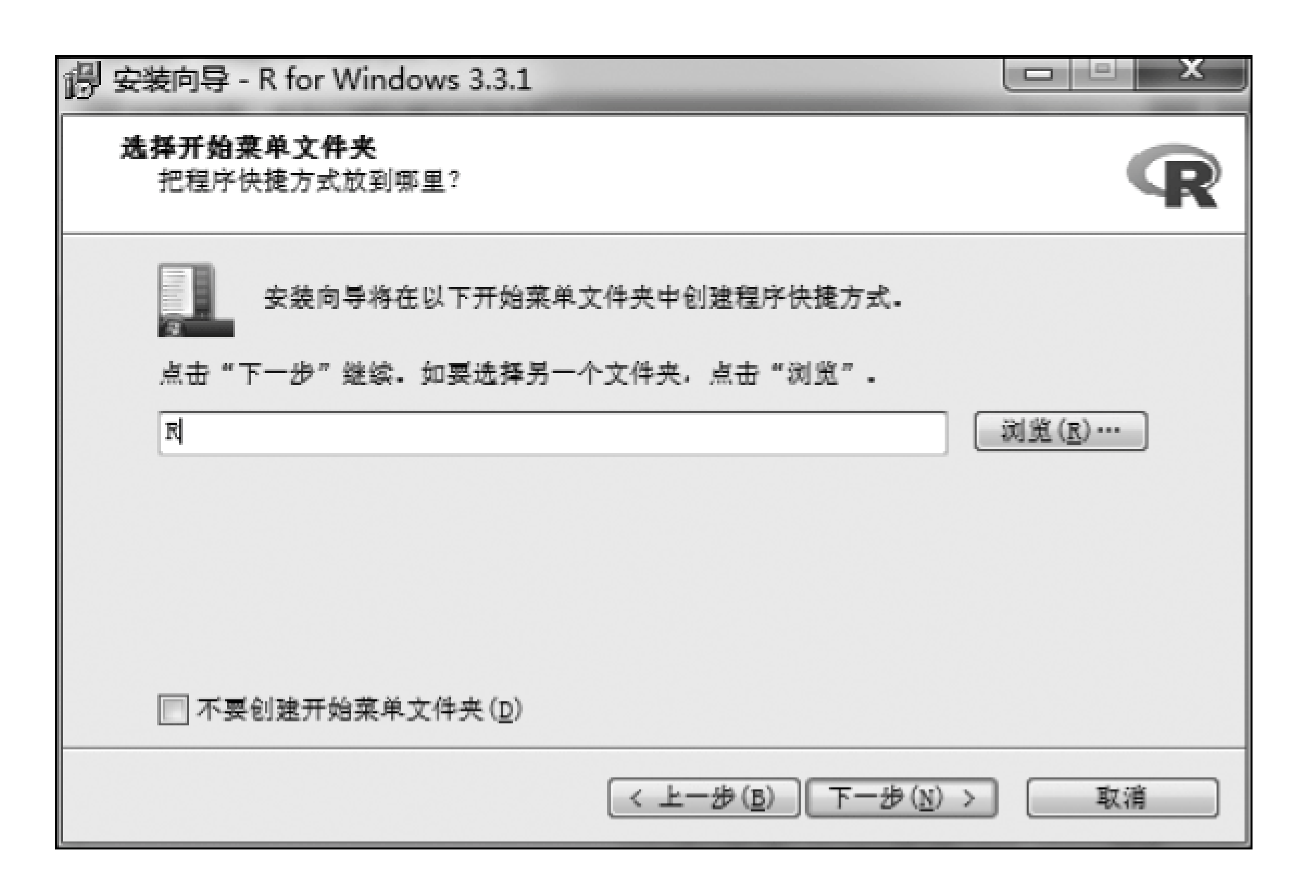


图 8-9 “选择开始菜单文件夹”界面

在“选择附加任务”对话框中，选择“创建桌面快捷方式”“版本信息保存于注册表内”和“把 R 程序同. RData 文件联合起来”复选框，单击“下一步”按钮，开始 R 的安装，如图 8-10 所示。

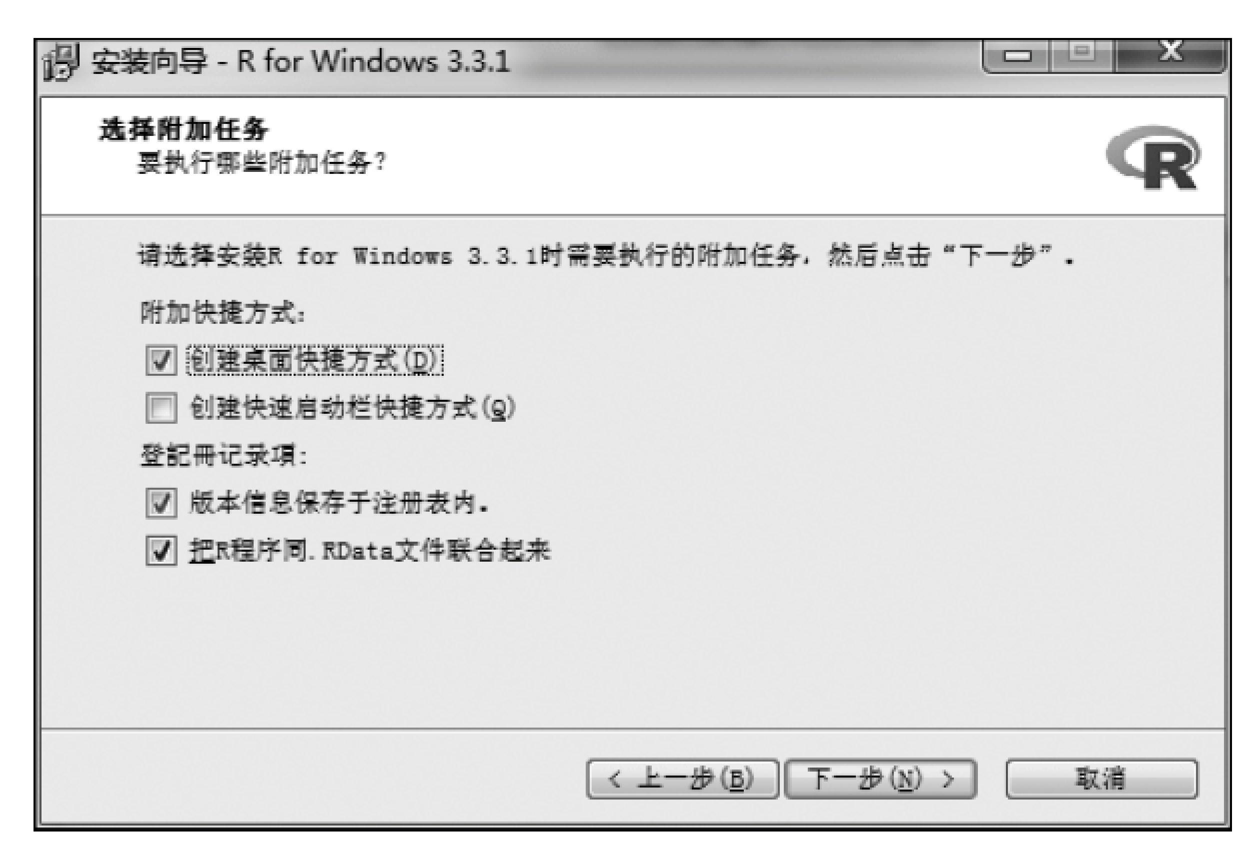


图 8-10 “选择附加任务”界面

R语言的安装会根据计算机性能不同，花费几十秒到几分钟时间，如图8-11所示。

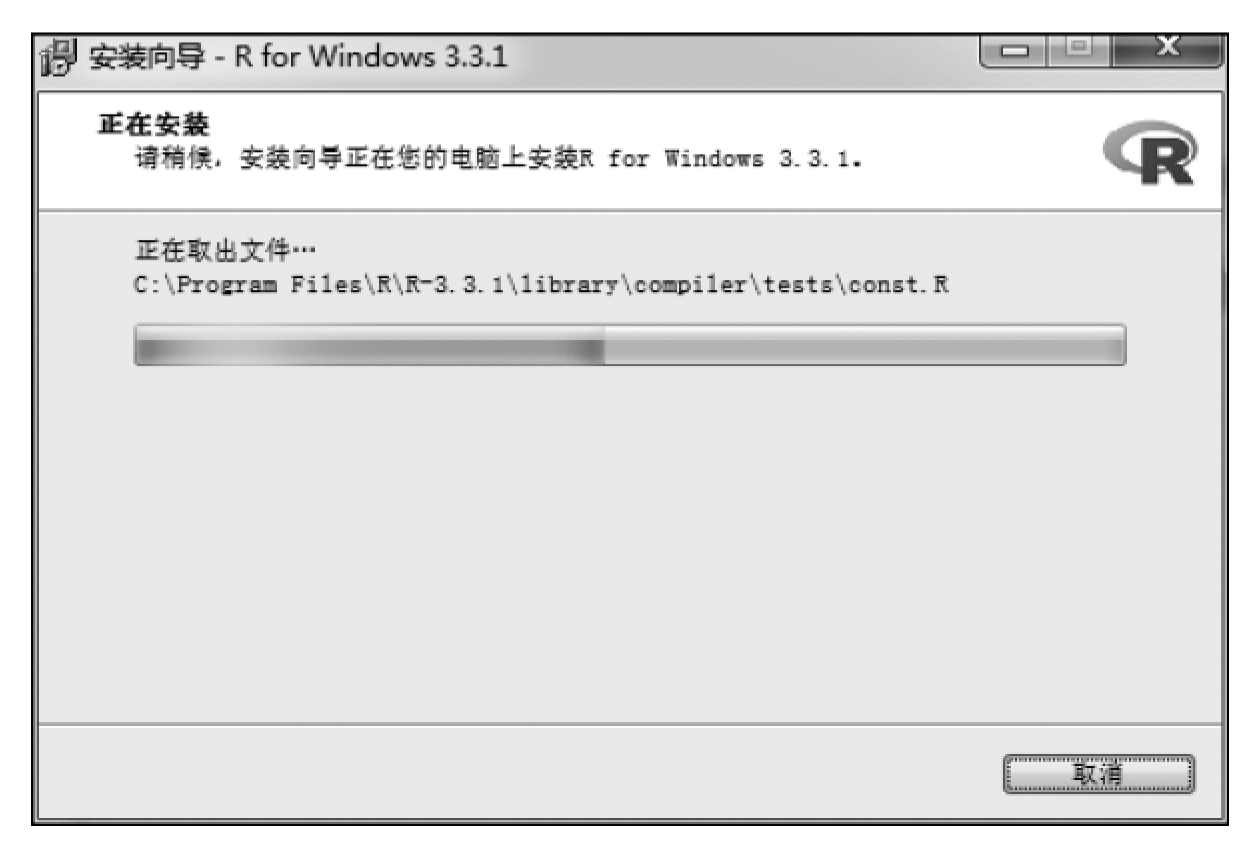


图8-11　正在安装R

当R程序安装完成后，会看到如图8-12所示的界面，单击"结束"按钮完成安装。选择"开始"→"所有程序"→R命令，可以看到R语言的选项，如图8-13所示。



图8-12　R语言安装完成



图8-13　R程序选项

8.2　使用R语言

本节给出R语言启动运行、R语言的常用操作及其包的使用，为读者使用R语言进行程序开发打下基础。首先描述R语言的启动运行方法，然后描述R语言的常用操作及日常命令，最后描述R语言包的使用方法。

8.2.1　运行R语言

选择"开始"→"所有程序"→R→R x64 3.3.1命令，启动R语言的RGui图形控制台，如图8-14所示。

与很多编程软件的界面相似，R语言的界面由菜单、快捷按钮和主窗口(R Console)

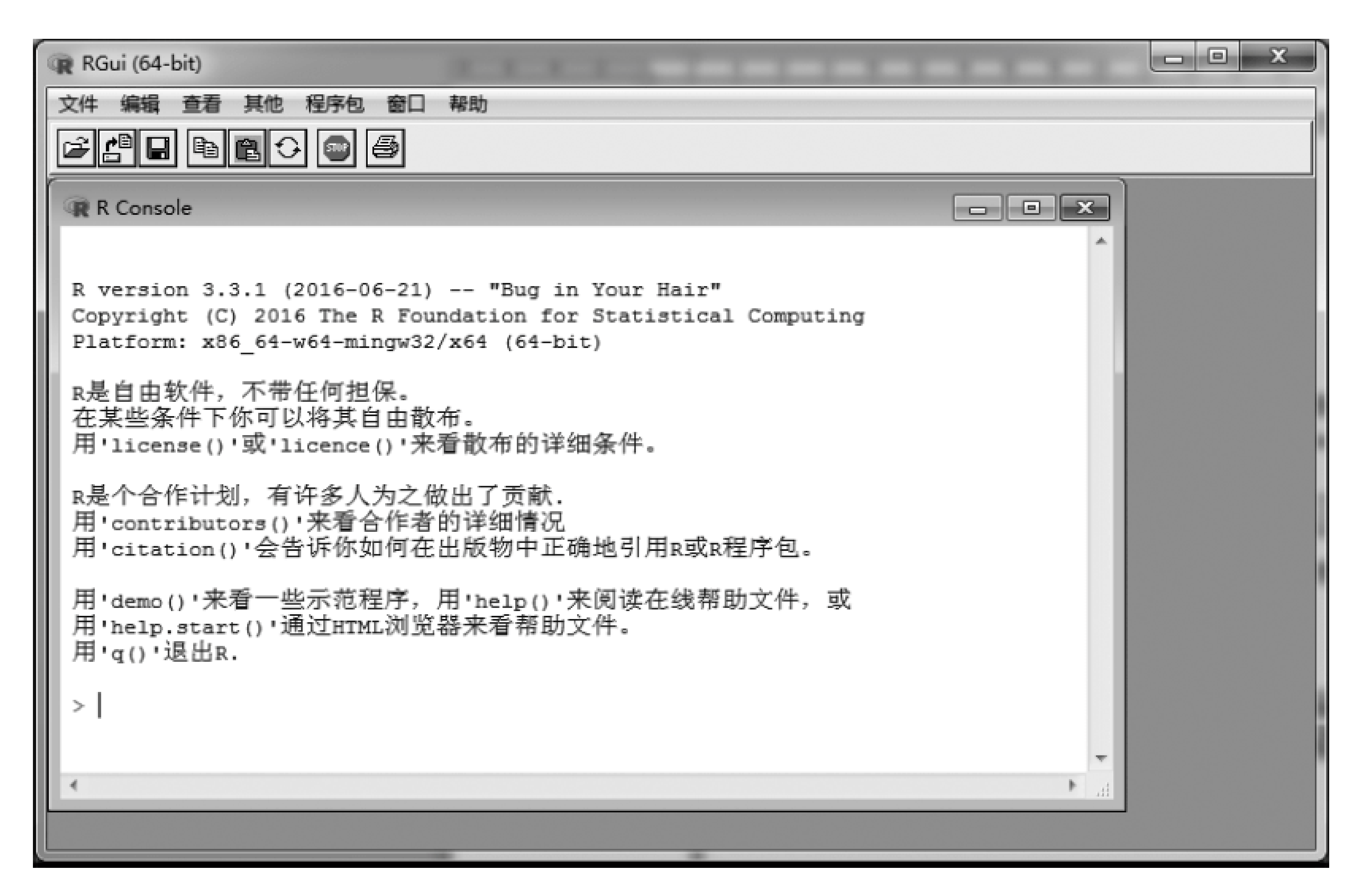


图 8-14　RGui 3.3.1 启动界面

组成。主窗口既是命令输入窗口，也是部分运算结果的输出窗口，有些图形的运算结果会在新建的窗口中输出。主窗口上方的文字是刚进入 R 语言环境的一些版本说明、软件授权介绍及指引，文字下的"＞"符号是 R 语言的命令提示符，在其后可输入命令。

在 R 语言简单而朴素的界面下，是丰富且复杂的运算功能和完善的绘图功能。最重要的一点是：R is free。这里的 free 不仅仅是免费，更强调"自由"软件，在遵循 GNU 相关授权许可协议的基础上，可以进行发布，在命令行运行 license()查看相关协议，如图 8-15 所示。

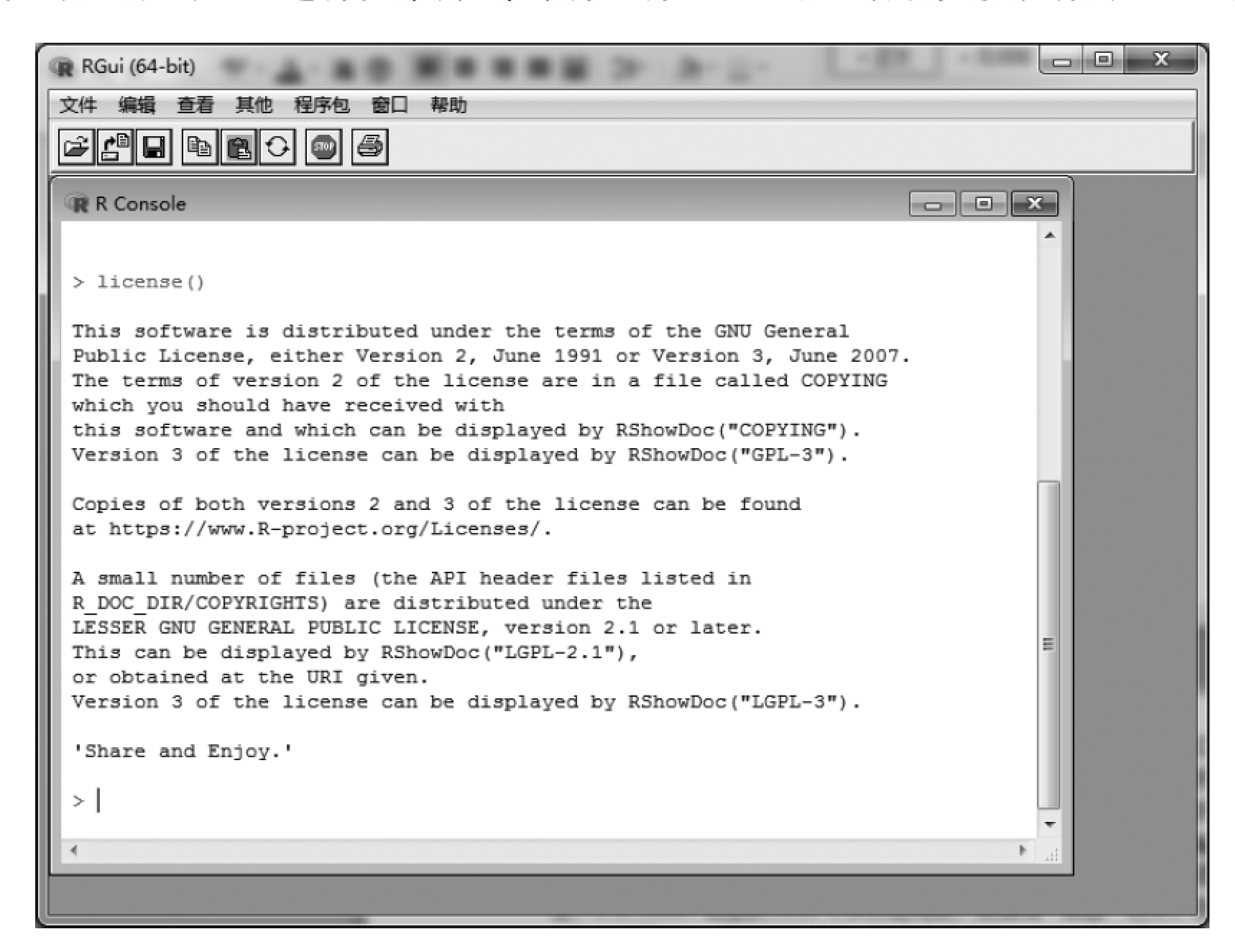


图 8-15　R 语言的 GNU 授权许可协议

8.2.2 R 语言常用操作

下面是 R 语言的 6 个常用操作。

1. Ctrl+ L 键

同时按下 Ctrl 和 L 键，清除命令窗口显示的所有内容，如图 8-16 所示。

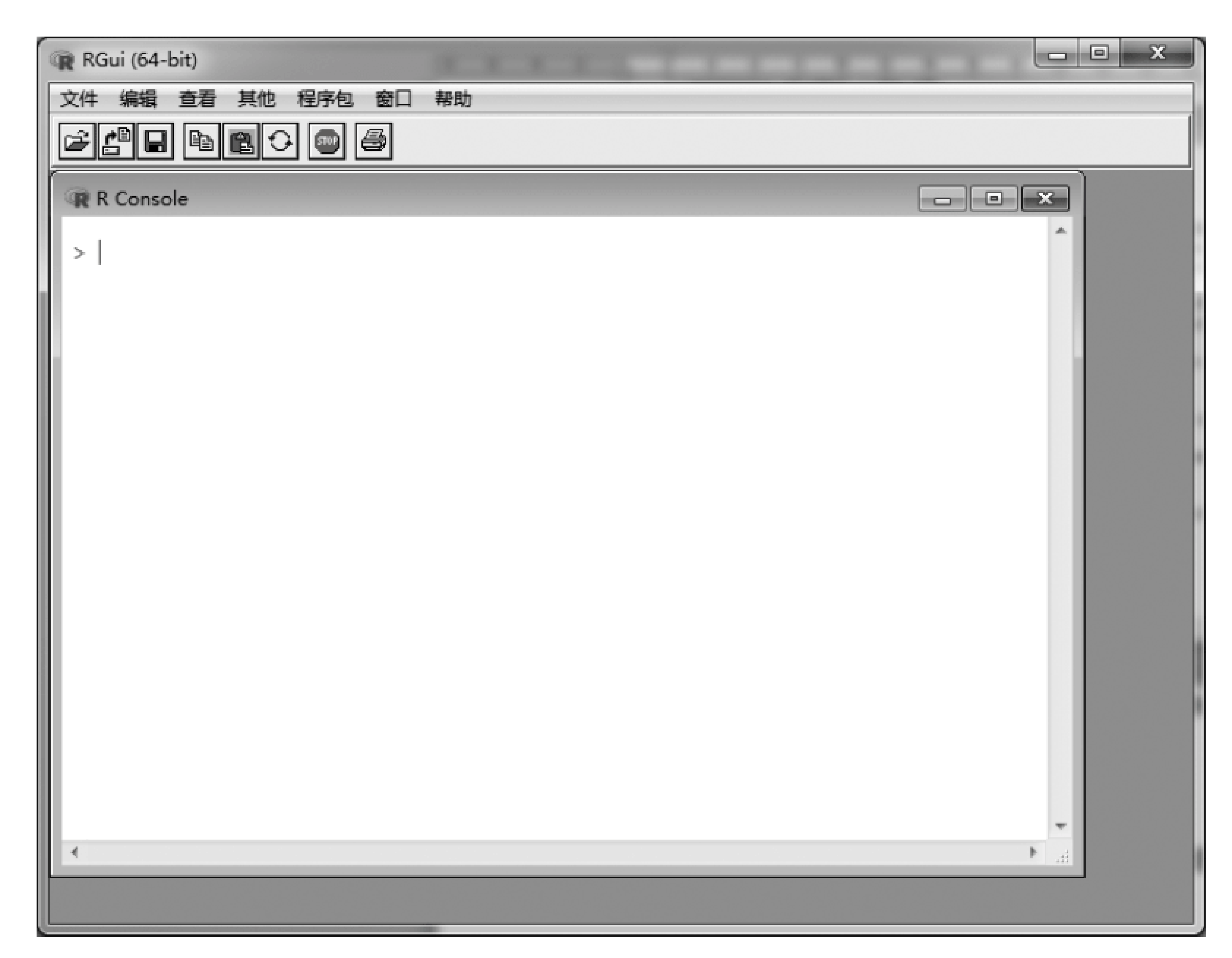


图 8-16　清除内容后的主窗口

2. help 命令

在命令窗口输入 help(函数名)，按回车执行，显示对应函数名的帮助信息。例如，要了解 demo 函数，可以在命令行输入 help(demo)，按回车执行，系统打开浏览器，并出现 demo 函数的相关内容，如图 8-17 所示。

在帮助窗口中主要包括如下 6 部分：

- Description，函数主要功能描述。
- Usage，函数调用格式及参数。
- Arguments，参数详细解释。
- Details，函数详细信息。
- See Also，参阅与本函数相关的其他函数链接。
- Examples，函数调用的例子。

3. ls 命令

在命令窗口输入 ls()，按回车执行，显示工作空间中的所有对象。工作空间是 R 语

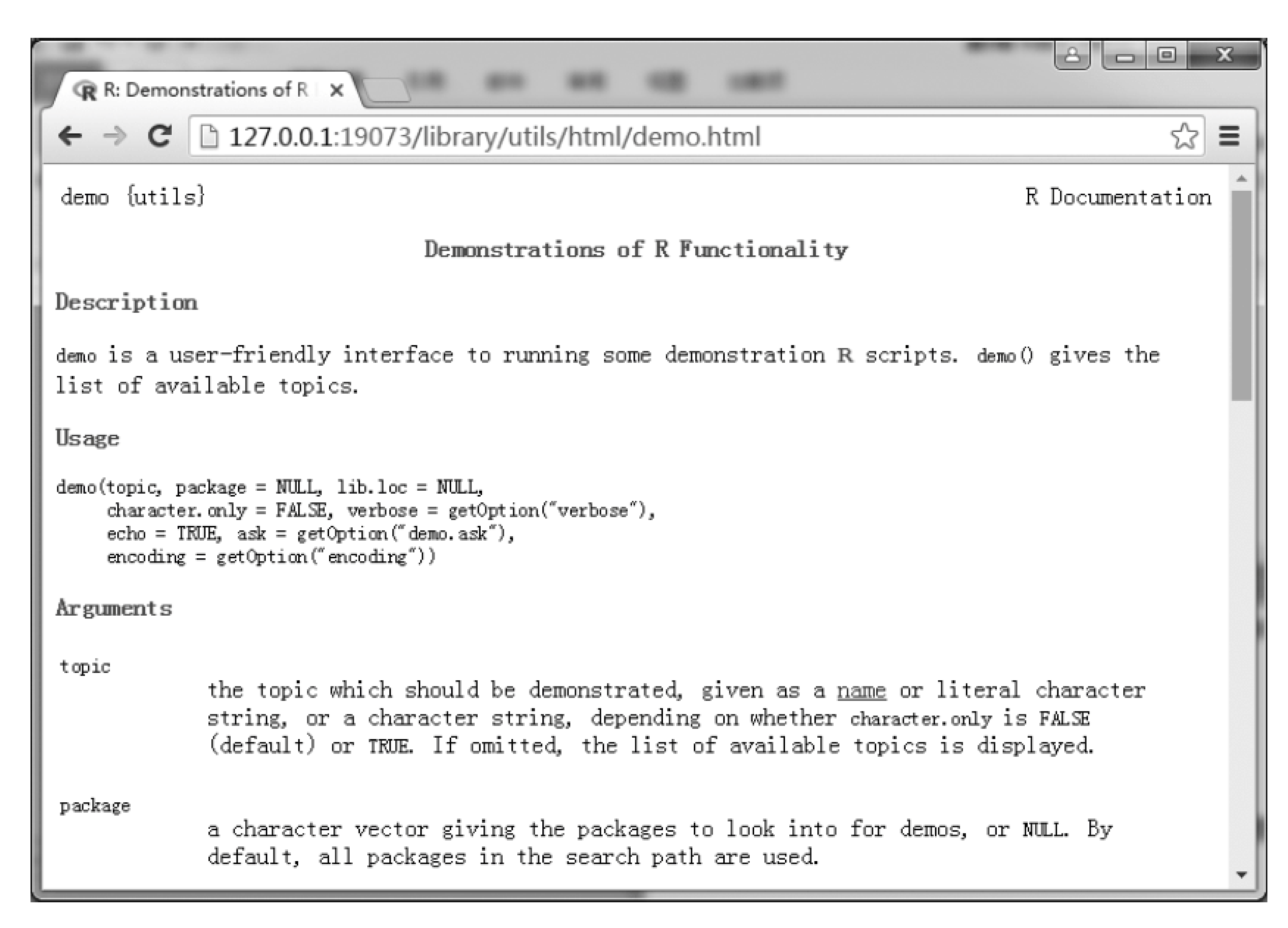


图 8-17 浏览器 demo 函数帮助窗口

言中一个非常重要的概念,因为在 R 语言中,所有的数据和函数都运行在内存中,这部分内存就是 R 语言的工作空间。

当任何函数与数据也没有运行的时候,运行 ls()后,显示 character(0),工作空间为空,如图 8-18 所示。

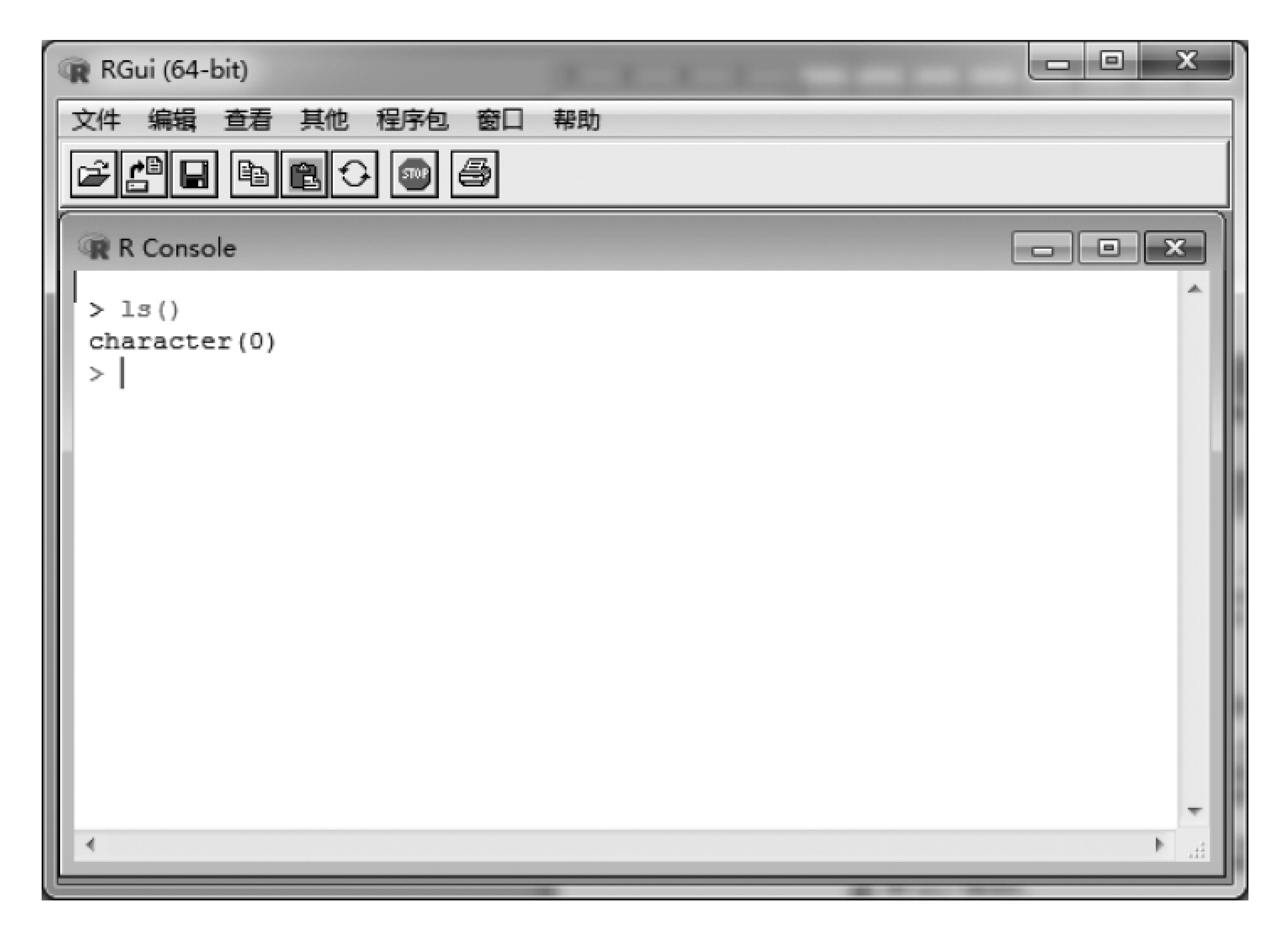


图 8-18 用 ls 查看为空的工作空间

在命令行运行一个简单的赋值语句 a<-10,把 10 赋值给变量 a,再运行 ls()查看工作空间,如图 8-19 所示。



图 8-19　用 ls 查看有值工作空间对象

4. rm 命令

在命令窗口输入 rm(对象名)，按回车执行，删除工作空间中的对象。也可以使用 rm(list=ls())来清除工作空间中的所有对象。

先运行 rm(a)删除工作空间中的对象；再运行 b<－"Test"给变量 b 赋值为字符串"Test"，执行 ls()命令，可以看到工作空间中有一个对象 b；再运行 c<－20 给变量 c 赋值 20，执行 ls()命令，可以看到工作空间中有两个对象 b 和 c；然后执行 rm(list=ls())清除所有对象；最后执行 ls()确认工作空间中的对象都被删除了。整个过程如图 8-20 所示。



图 8-20　用 rm 删除工作空间对象

5. getwd 命令

在命令窗口输入 getwd()，获取当前工作目录。工作目录是 R 语言中另外一个非常重要的概念，工作目录表示当前在哪个文件夹下工作。在存取文件的时候，如果不指定路径，就会默认在这个文件夹下进行，如图 8-21 所示。

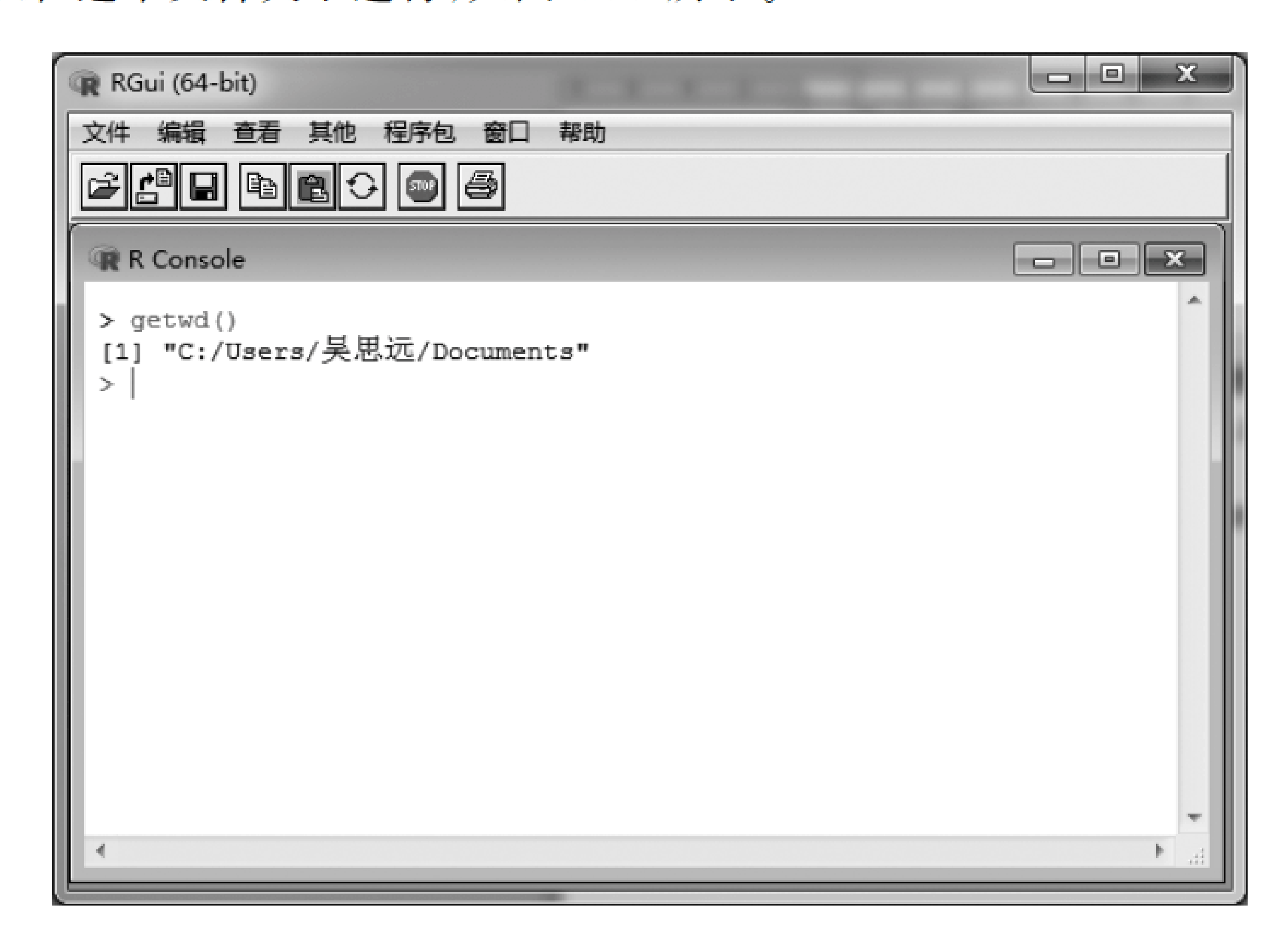


图 8-21　用 getwd 获取工作目录

6. setwd 命令

在命令窗口输入 setwd(目录)，设置工作目录到指定的路径。在命令行执行 setwd("D:/快盘/2015—2016 第二学期/大数据智能分析与处理")，如图 8-22 所示。



图 8-22　用 setwd 设置工作目录

8.2.3 包的使用

R 语言的一个重要特点是其拥有不同领域的专业人士编写的软件加载包，大部分包可以从 CRAN 和 Bioconductor 上找到。每一个包都是一系列函数的集合，可能还带有数据、说明文档、源代码等。那么在 CRAN 上到底有多少加载包可以供我们使用呢？可以在命令行中执行 ap<－names(available. packages()[,1])，把可以使用的加载包赋值给变量 ap，在连线选择 CRAN 镜像服务器时，选择 China，再运行 ap 显示加载包名称和数量，如图 8-23 所示。

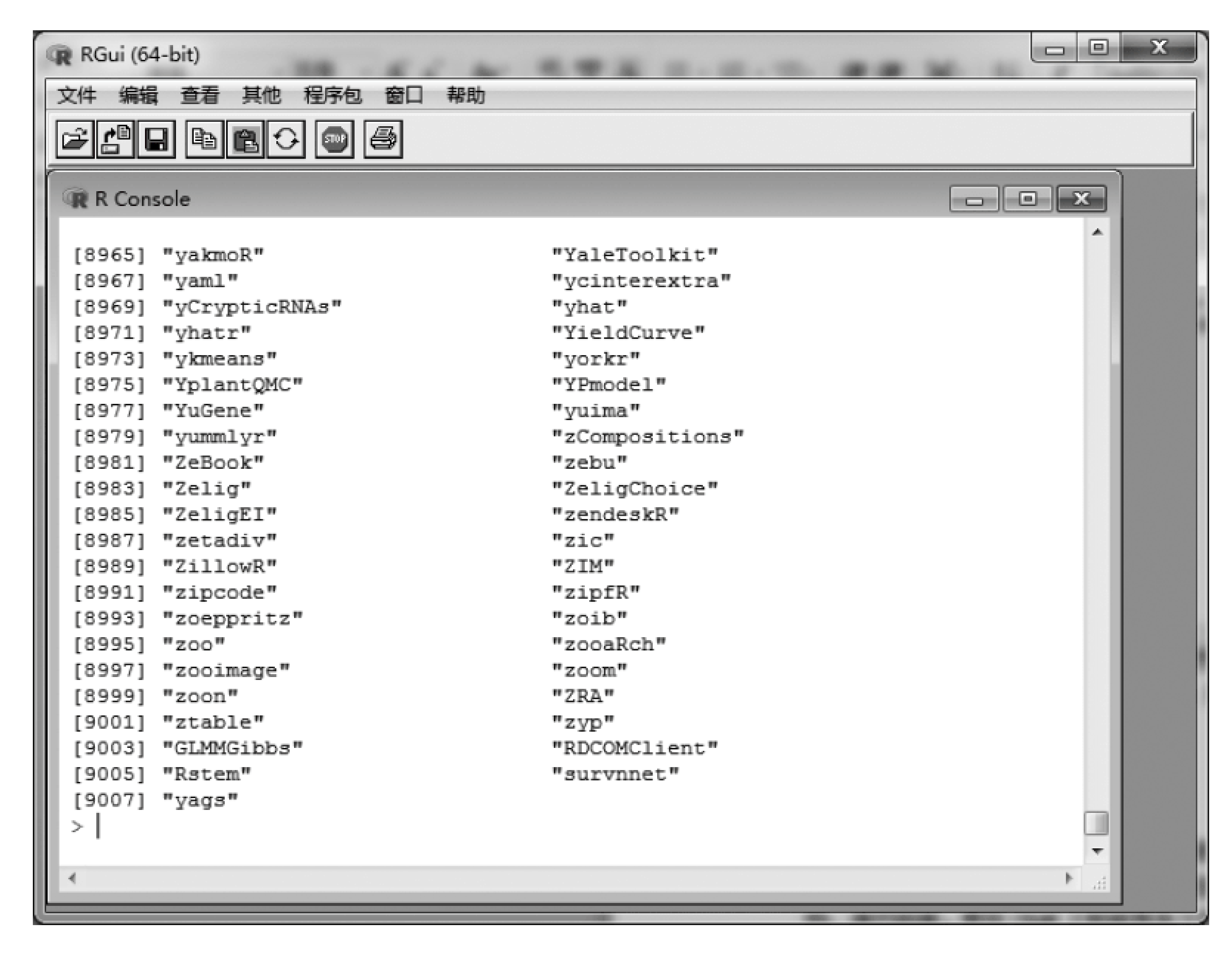


图 8-23　查看 CRAN 上可用加载包

截至 2016 年 8 月 30 日，CRAN 上可以加载的包有 9007 个。这个数量是相当惊人的。如果再加上 Bioconductor 上可以加载的包，软件加载包的总量至少在 1 万以上。如果要使用这些加载包，首先要安装，在使用前要加载。

1. 包的下载和安装

在命令窗口输入 install. packages("安装的包名称"，dependencies＝TRUE)，完成包的下载和安装。在命令行执行 install. packages ("RODBC"， dependencies＝TRUE)，下载安装 R 语言的数据库连接包。如图 8-24 所示。

为了检查 RODBC 数据库连接包是否下载安装成功，可以在命令行执行 installed. packages()[,c("Package","Version")]，查看所有已经安装的包，图 8-25 中矩形框圈出的部分为已安装好的 RODBC 包。



图 8-24 安装 RODBC 加载包

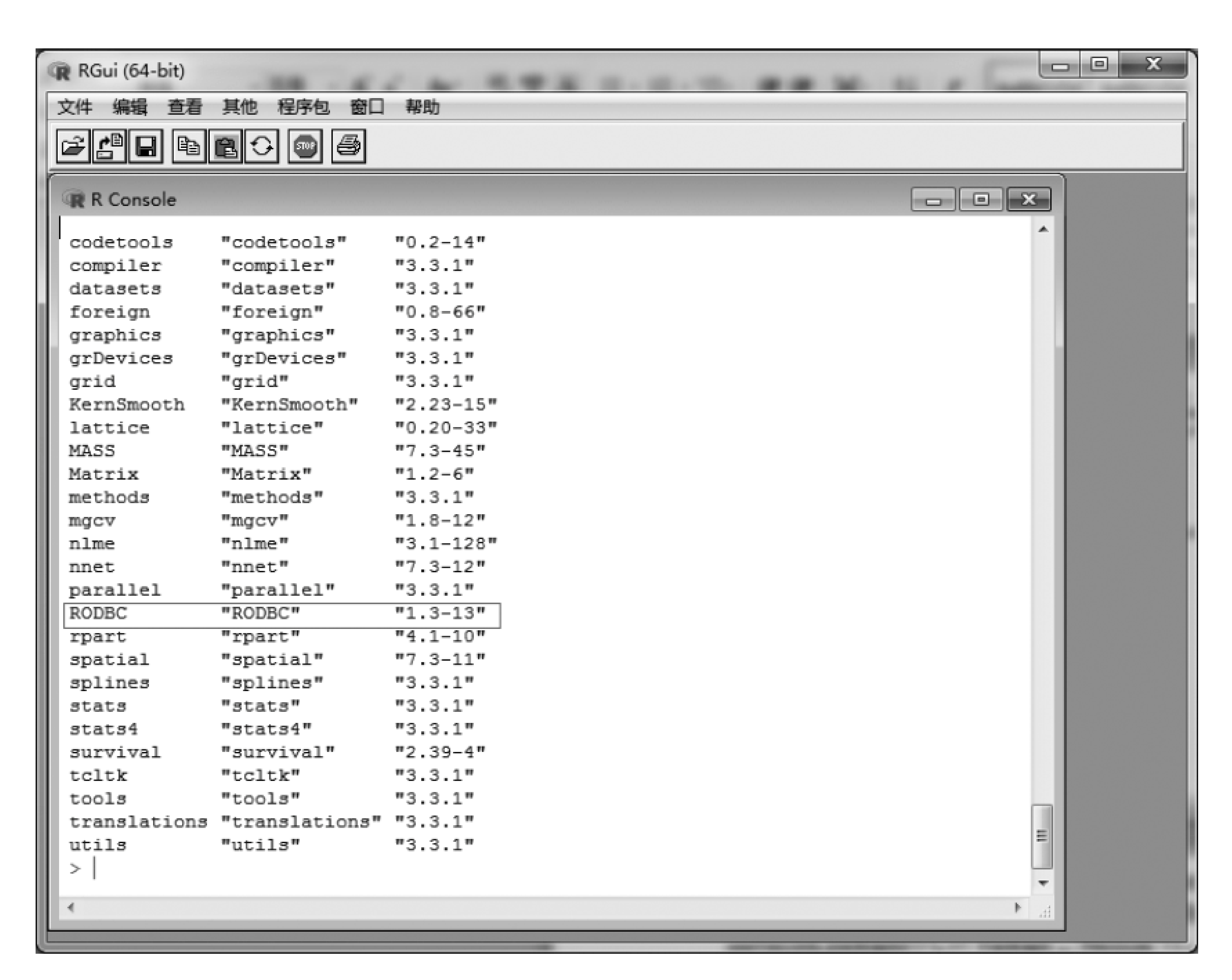


图 8-25 查看所有已经成功安装的包

2. 包的加载

在命令窗口输入 library("加载的包名称")，完成包的加载，加载后便可以使用包中的相关函数。在命令行执行 library("RODBC")，加载 R 语言的数据库连接包，如图 8-26 所示。

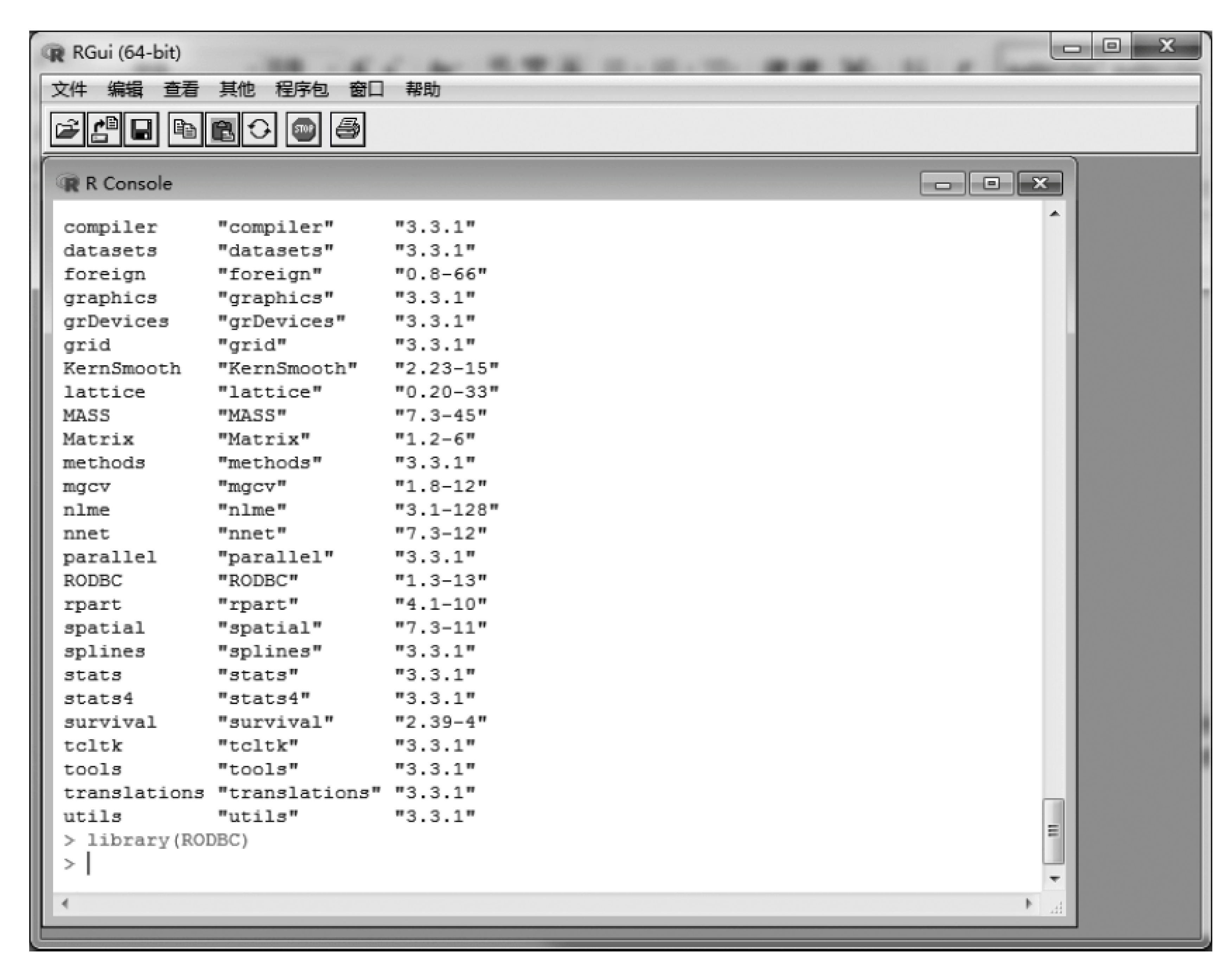


图 8-26　加载 RODBC 包

3. 卸载内存中的包

在命令窗口输入 detach (package："包名称")，完成内存中包的卸载，卸载后无法继续使用包中的相关函数，但释放相应的内存空间给操作系统。在命令行执行 detach (package:RODBC)，从内存中卸载 R 语言的数据库连接包。

8.3　R语言的数据结构

R 语言的数据结构主要与线性代数中的一些概念相似，如向量、矩阵等。值得注意的是，R 语言中其实没有如数值型、字符型、逻辑型等的简单数据类型，对于简单类型 R 语言会自动看做长度为 1 的向量。R 语言中最重要的数据结构是向量、矩阵、数组、因子、列表和数据框。本节详细介绍 R 的向量、矩阵、数组。因子、列表、数据框。

8.3.1　向量

向量是 R 语言最基本的数据结构，用来存储一维数据，其中的数据必须具有相同的

数据类型，比如都是字符或都是整数，可以用函数 c()来创建向量。例如：

```
var_number <-c(2,4,6,8,10)
var_character <-c("CQUPT","Chongqing","China")
var_boolean <-c(TRUE,TRUE,FALSE,FALSE)
```

在 R 语言中可以使用等号“＝”来赋值，也可以使用箭头“<－”来赋值，两种方法几乎没有区别，但一般在赋值的时候使用箭头，传递参数的时候使用等号。

使用方括号来访问向量中的元素，在 R 中，向量下标不是从 0 开始的，而是从 1 开始的。例如，访问向量 var_number 的第 1 个元素，用 var_number[1]；访问向量 var_number 的第 3 到第 5 个元素，用 var_number[3:5]；访问向量 var_number 中除了第二个元素以外的其他所有元素，用 var_number[－2]，如图 8-27 所示。



图 8-27　创建和访问向量

8.3.2　矩阵

矩阵是一个二维的数组，与向量一样，其中的数据应具有相同的类型。在 R 语言中使用 matrix()函数来创建矩阵，matrix()的用法为：

```
matrix(data=NA, nrow=1, ncol=1, byrow=FALSE, dimnames=NULL)
```

其中，data 表示一个向量，即需要创建的矩阵中的元素；nrow 表示行数；ncol 表示列数；byrow 是一个逻辑变量，默认为 FALSE，表示按列排列，如果为 TRUE，表示按行排列；dimnames 表示向量的行和列的名字，默认没有名字。

用如下语句创建一个 2 行、3 列，按行排列，元素从 1 到 6 的矩阵 m1：

```
m1 <-matrix(data=c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=TRUE,
    dimnames=list(c("row1", "row2"),c("col.1", "col.2", "col.3")))
```

与向量类似，可以通过下标来取出矩阵中的元素，如 m1[1,3]表示第一行第三列的元

素，而 m1[2,]表示第二行元素。结果如图 8-28 所示。

```
RGui (64-bit)
文件 编辑 查看 其他 程序包 窗口 帮助
R Console
> m1 <- matrix(data = c(1,2,3,4,5,6), nrow = 2, ncol = 3, byrow = TRUE,
+                  dimnames = list(c("row1", "row2"),
+                                  c("col.1", "col.2", "col.3")))
> m1
     col.1 col.2 col.3
row1     1     2     3
row2     4     5     6
> m1[1,3]
[1] 3
> m1[2,]
col.1 col.2 col.3
    4     5     6
> |
```

图 8-28　创建和访问行矩阵 m1

用如下语句创建一个 2 行、3 列，按列排列，元素从 1 到 6 的矩阵 m2：

```
m2 <-matrix(data=c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=FALSE,
    dimnames=list(c("row1", "row2"),c("col.1", "col.2", "col.3")))
```

m2[,2]表示第二列元素，如图 8-29 所示。

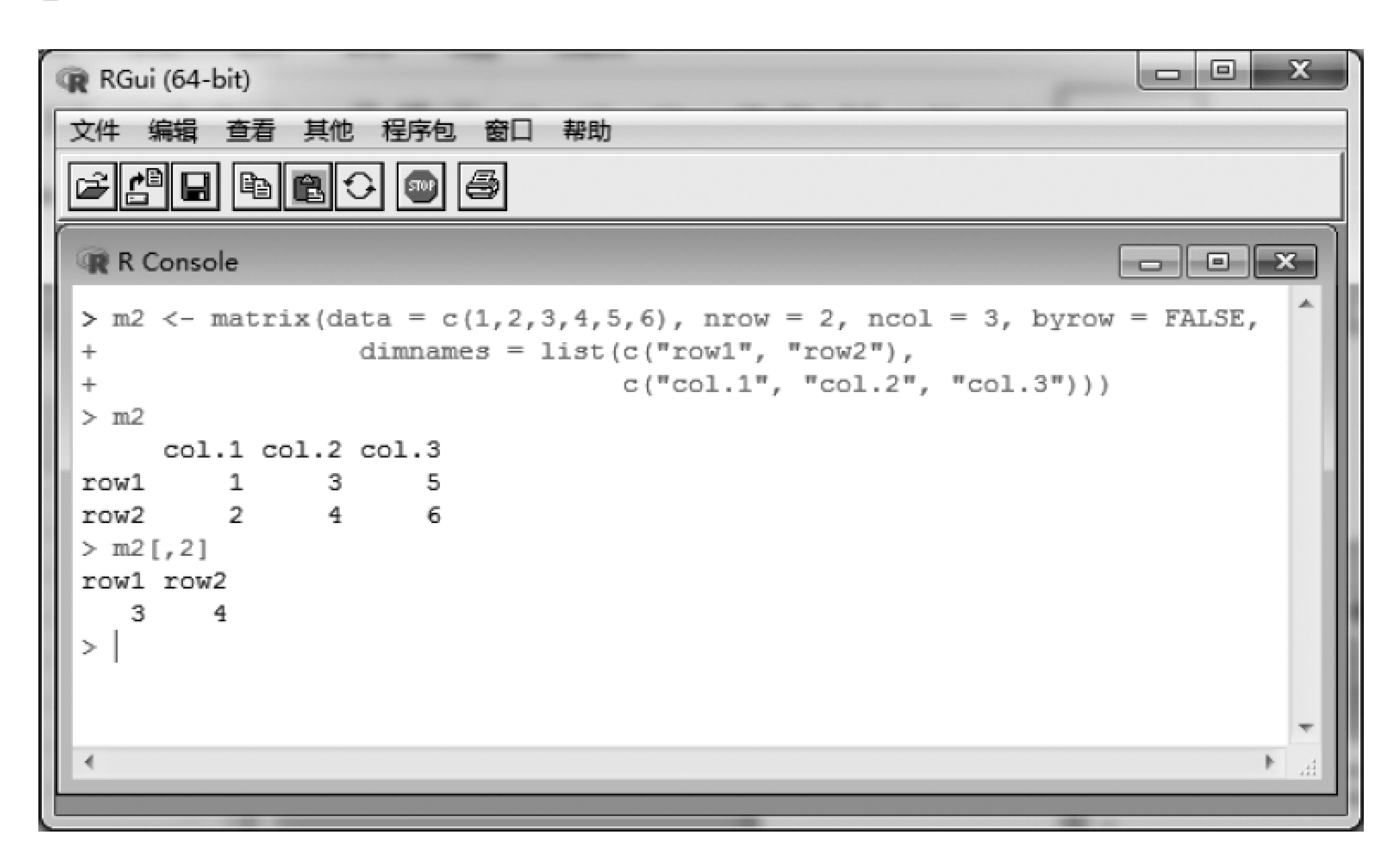


图 8-29　创建和访问列矩阵 m2

8.3.3　数组

数组是带有多个下标的类型相同的元素的集合，数组中元素的类型也要相同，可以通过 array()函数来创建数组，array()的用法为：

```
array(data=NA, dim=length(data), dimnames=NULL)
```

其中,data 表示一个向量,即需要创建的数组中的元素;dim 表示数据的维度,默认值是数据的长度;dimnames 用来指定各个维度的名字,默认没有名字。

用如下语句创建一个三维的数组 ar1,

```
ar1 <- array(data=c(1:4,11:14,21:24), dim=c(2, 2, 3),dimnames=list(c("dim1.1",
    "dim1.2"), c("dim2.1","dim2.2"),c("dim3.1","dim3.2","dim3.3")))
```

ar1[1,1,1]表示维度 1、2 和 3 都在第一个的元素,ar1[2,1,2] 表示维度 1 的第二个、维度 2 的第一个、维度 3 的第二个元素,如图 8-30 所示。

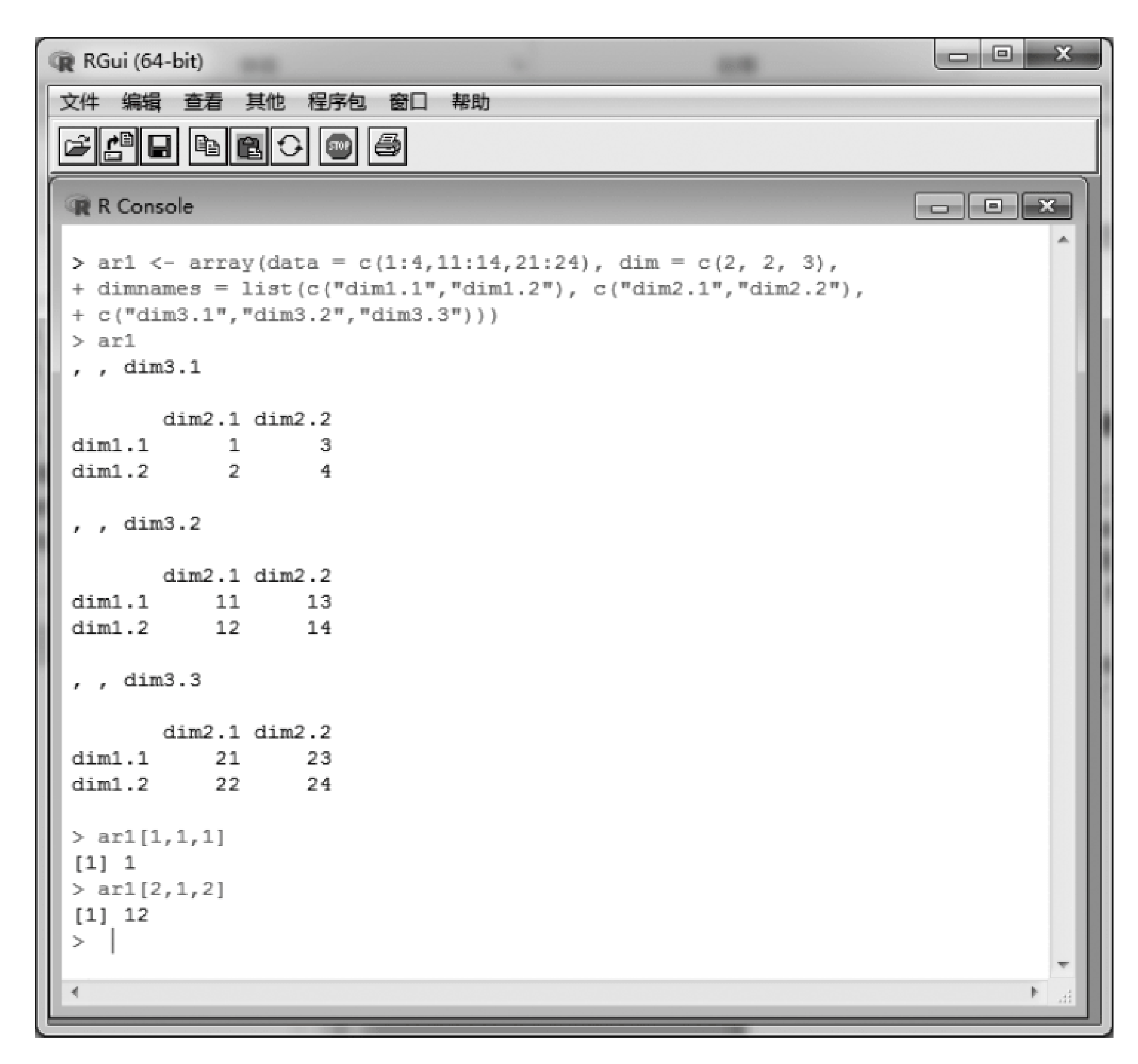


图 8-30　创建和访问数组 ar1

可以尝试着改变下标,看看取出的值是否与预期的一样,加深对 R 语言数组的理解。

8.3.4　因子

因子用来表示分类变量,这类变量不能用来计算,而只能用来分类或者计数,如性别或者民族等,使用 factor()函数来创建因子,factor()的用法为

```
factor(x=character(), levels, labels=levels,exclude=NA,
    ordered=is.ordered(x), nmax=NA)
```

其中,x 表示创建因子所需的向量;levels 表示创建因子向量级别,如果不指定,就是 x 中所有值都不重复;labels 是级别标签,与前面的 levels 一一对应,便于识别;exclude 表示哪些级别需要排除在外;ordered 表示因子是有序因子还是无序因子,为 TRUE 时,x 为有序因子,为 FALSE 时,x 为无序因子;nmax 表示级别数量的上限值。

用如下语句创建一个向量，并用向量创建 3 个因子，通过 3 个因子的创建和显示，可以很好地了解 levels、labels 等参数，如图 8-31 所示。

```
color <-c('R', 'G', 'Y', 'Y', 'G', 'R', 'Y', 'R', 'Y')
fa1 <-factor(color)
fa1
fa2 <-factor(color, levels=c('G', 'R', 'Y'), labels=c('绿', '红', '黄'))
fa2
fa3 <-factor(color, levels=c('G', 'R' ), labels=c('Green', 'Red'))
fa3
```



图 8-31　创建和访问因子

因子 fa3 中显示为 NA 的表示值不可用(not available)或者值缺失。

8.3.5　列表

前面学习的几种数据结构都只能存储同一类型的值，如果一个数据结构需要包含不同的数据类型，则可以采用列表这种形式。使用 list()函数来创建列表，list()的用法为

```
list(name1=component1, name2=component2, …)
```

其中，name 和 component 分别表示列表中的元素的名称与值。用如下语句创建一个包含 3 个因子的列表，并使用[]和 $ 对列表进行访问，如图 8-32 所示。

```
id <-c(1,2,3)
title <-c('Mr A', 'Mr B', 'Mr C')
depart <-c('Computer Science','Information Security','Geographic Information')
teacher <-list(group=id, name=title, decription=depart)
teacher
teacher["group"]
teacher$ name
teacher[3]
```

```
teacher$ n
teacher["n"]
```

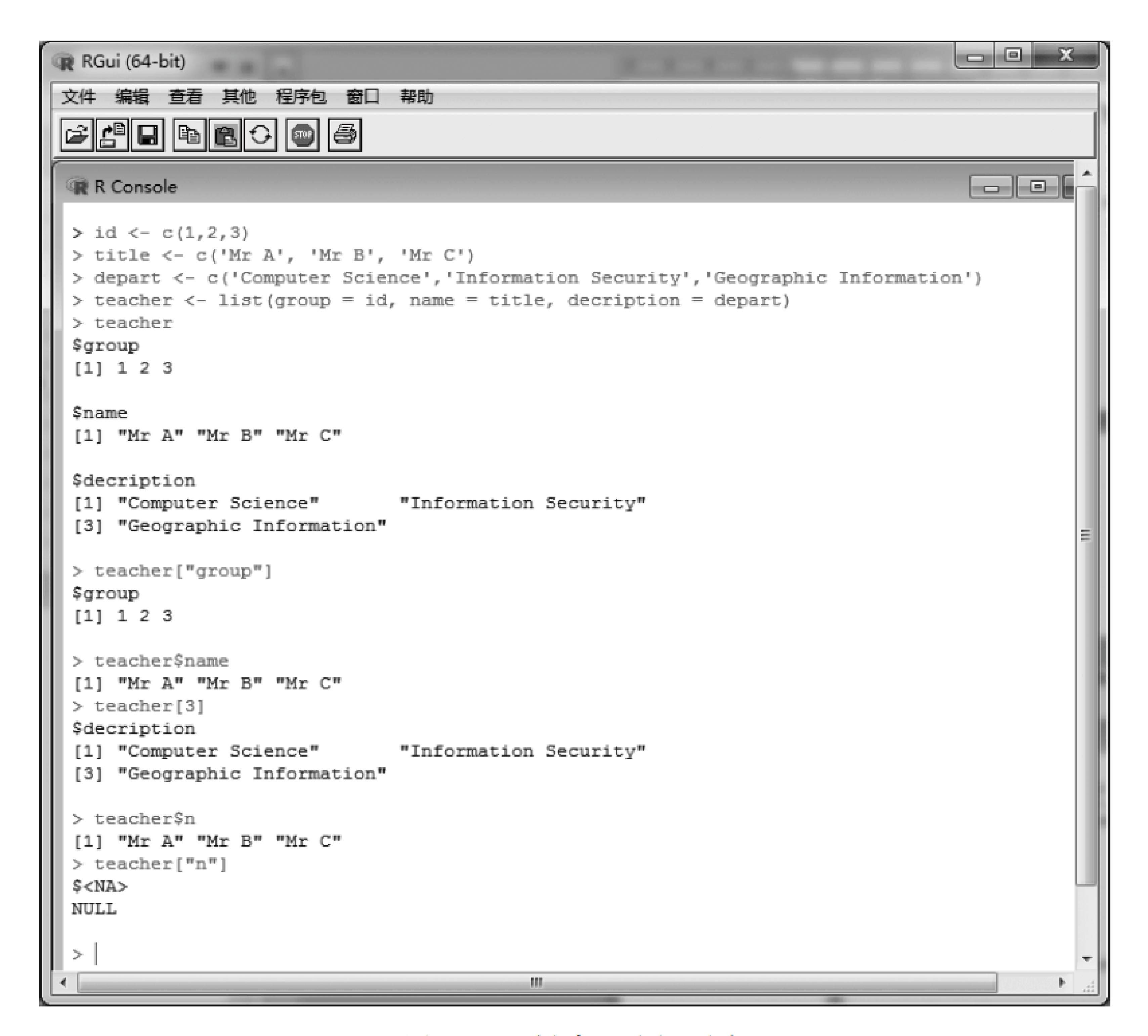


图 8-32　创建和访问列表

使用[]和$取值的区别如下：[]使用精确值来匹配；而$使用模糊匹配，只要没匹配到重复的，就可以取到值。这就是 teacher$n 能取到值，而 teacher["n"]取不到值的原因。

8.3.6　数据框

R 语言的数据框，也称为数据帧，是 R 语言中使用非常广泛的一种数据结构，在进行数据分析的时候，通常会使用数据框来进行数据的存储，有点类似微软公司的 Excel 电子表格。可以把数据框理解为可以在各列存放不同数据类型元素的矩阵，当然每一列的数据类型要求一样，整个数据框各列之间的数据类型可以不同。使用 data.frame()函数来创建列表，data.frame()的用法为

```
data.frame(…, row.names=NULL, check.rows=FALSE,check.names=TRUE,
fix.empty.names=TRUE,stringsAsFactors=default.stringsAsFactors())
```

其中，“…”表示各个列的数据集；row.names 表示行的名称；check.rows 是逻辑值，表示是否检查行的长度和名称；check.names 也是逻辑值，表示检查数据框中的变量名称是否合法以及是否有重复；fix.empty.names 也是逻辑值，表示如果参数是未命名的，是否可以自动生成一个名称；stringsAsFactors 也是逻辑值，表示是否把数据集中的字符串转化

为因子。

用如下语句创建一个包含 4 个列的数据框，并用[]对数据框进行访问，如图 8-33 所示。

```
var1 <- c(101,102,103,104,105)
var2 <- c(25,22,29,34,33)
var3 <- factor(c("male","male","female","female","male"))
var4 <- c("张三","李四","王五","赵六","钱七")
user.dat <- data.frame(id= var1, age= var2, gender= var3,name= var4)
user.dat
user.dat[1,]
user.dat[,2]
user.dat[3,3]
user.dat[4,4]
```

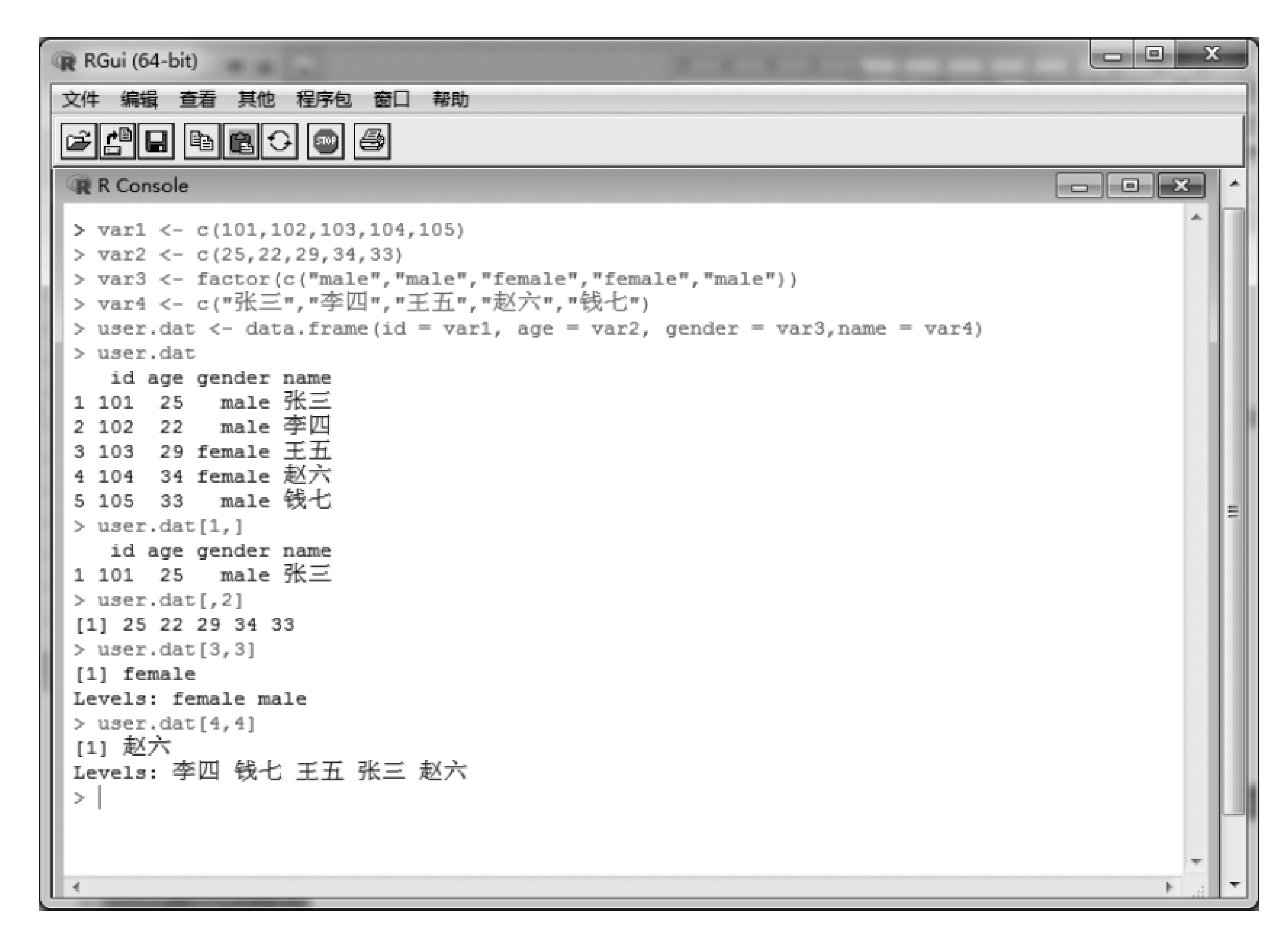


图 8-33　创建和访问数据框

8.4　R语言的编程结构

R 语言是一种表达式语言，也就是说其命令类型只有函数或表达式，并由它们返回一个结果，与 C、C++、Python、Perl 等语言类似，也有编程结构。本节介绍 R 语言的条件语句和循环语句。

8.4.1　条件语句

R 语言的条件语句主要有 if-else 语句、ifelse 语句和 switch 语句。

if-else 语句的语法为

```
if (condition) {expr1} else {expr2}
```

expr1、expr2 可以为一个或一组语句，若只有一个语句，可省略大括号。

if-else 语句可以进行如下的嵌套使用：

```
if (condition1) {expr1} else if (condition2) {expr2} else {expr3}
```

看如下两个简单的例子以了解 if-else 语句的用法，其中 cat 和 print 为屏幕打印函数，结果如图 8-34 所示。

```
x <-8
if (x<0) cat ("x小于零")
if (x>=0) print(x) else print(-x)

y <--10
if (y<0) cat ("y小于零")
if (y>=0) print(y) else print(-y)
```



图 8-34 if-else 语句运行结果

ifelse 语句的语法为

```
ifelse(condiction,yes,no)
```

ifelse 语句对 condition 进行逻辑判断，如果为 TRUE，就执行 yes 语句，否则执行 no 语句。

看如下的例子以了解 ifelse 语句的用法，结果如图 8-35 所示。

```
a <-c(3:-3)
ifelse(a>=0,a,-a)
```

switch 语句的语法为

```
switch(expr,…)
```

其中,expr 为表达式,若其返回值在 1 至 length(…)之间,则返回列表相应位置的值,否则返回“NULL”;“…”为列表。若“…”的列表对象有名称,则 expr 表达式等于变量名时,返回该变量名对应的值。

看如下的例子以了解 switch 语句的用法,其中函数 mean(1:10)是计算 1+2+…+10 的平均值是多少,结果如图 8-36 所示。

```
mean(1:10)
switch(2,mean(1:10),1:5,1:10)
```

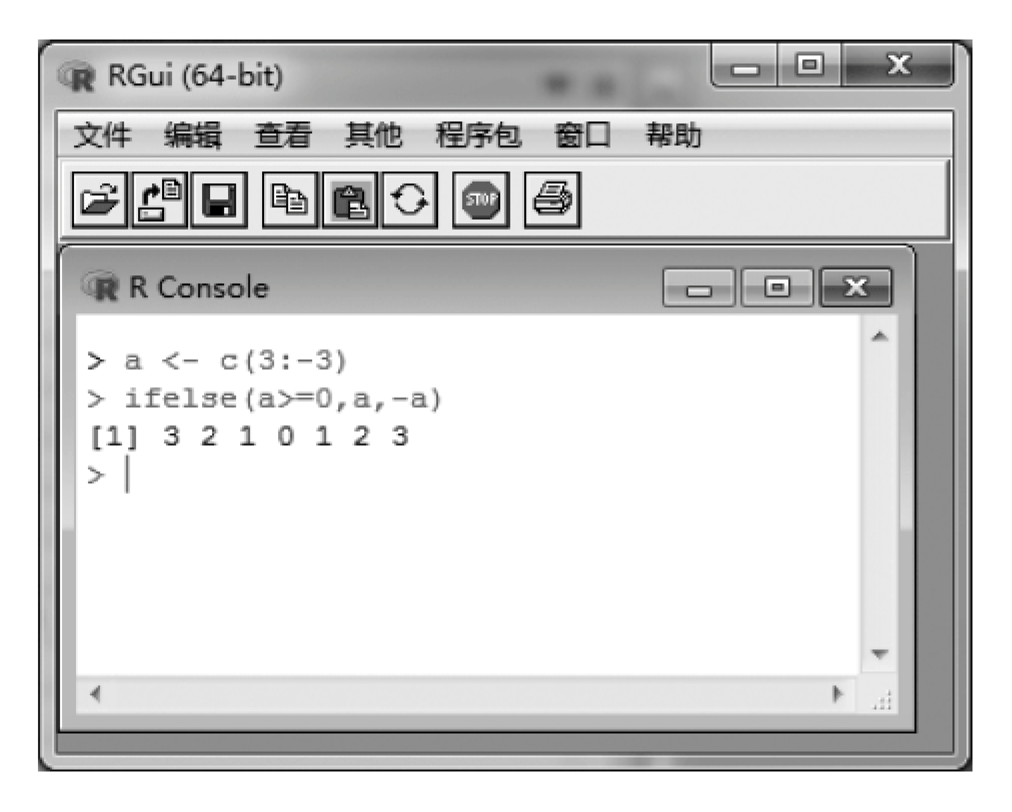


图 8-35 ifelse 语句运行结果

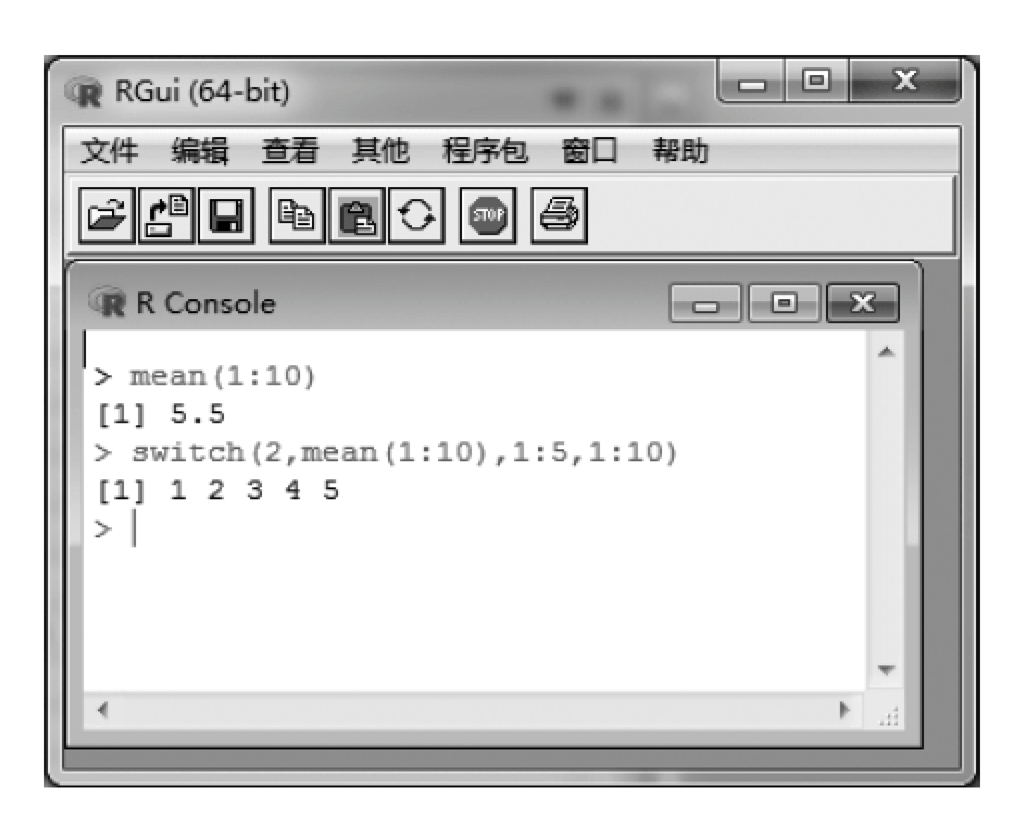


图 8-36 switch 语句运行结果

8.4.2 循环语句

R 语言的循环语句主要有 for 语句、while 语句和 repeat 语句。

for 语句的语法为

```
for(i in seq) {expr}
```

其中,i 为循环变量;seq 一般为序列,每次循环时 i 依次从 seq 中取值;expr 为一个或一组表达式,当 i 在 seq 中时,则执行 expr 的语句,否则循环终止。

循环过程中,若要输出每次循环的结果,可使用函数 cat()或 print(),cat( )的基本格式为

```
cat(expr1, expr1, …)
```

其中,expr1 和 expr2 为要输出的内容,可以是字符串(输出该字符串)或表达式(输出表达式的值)。若要换行输出,使用“\n”。

看如下的例子以了解 for 语句的用法,其中函数 sqrt 是求平方根函数,结果如图 8-37 所示。

```
n<-c(4,9,16,35)
  for(i in n) {
```

```
x<-sqrt(i)
cat("sqrt(", i, "): ", x, "\n")
}
```

while 语句的语法为

```
while(condition) {expr}
```

其中，condition 为判断条件，expr 为一个或一组表达式。while 循环重复执行 expr 的语句，直到条件 condition 不为真为止。

看如下的例子以了解 while 语句的用法，结果如图 8-38 所示。

```
x <-2
while ( x<=10)
{
print(x)
x=x+1
}
```

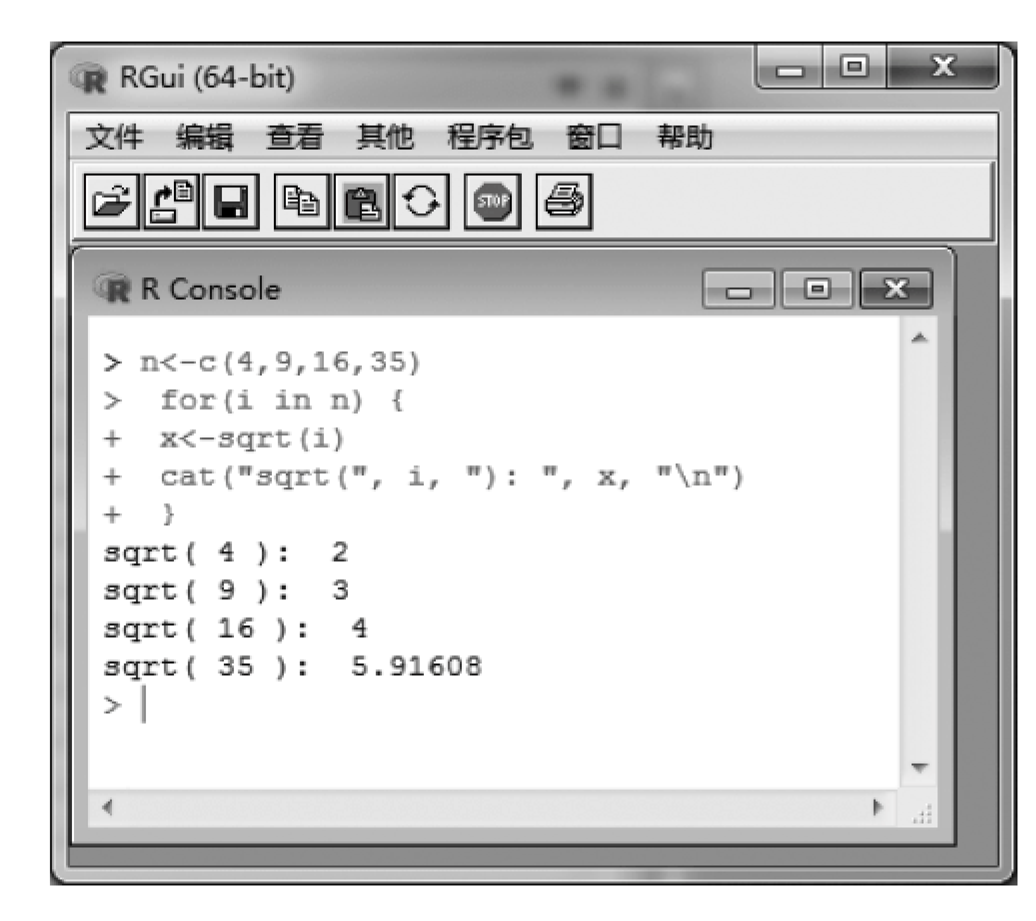


图 8-37 for 语句运行结果

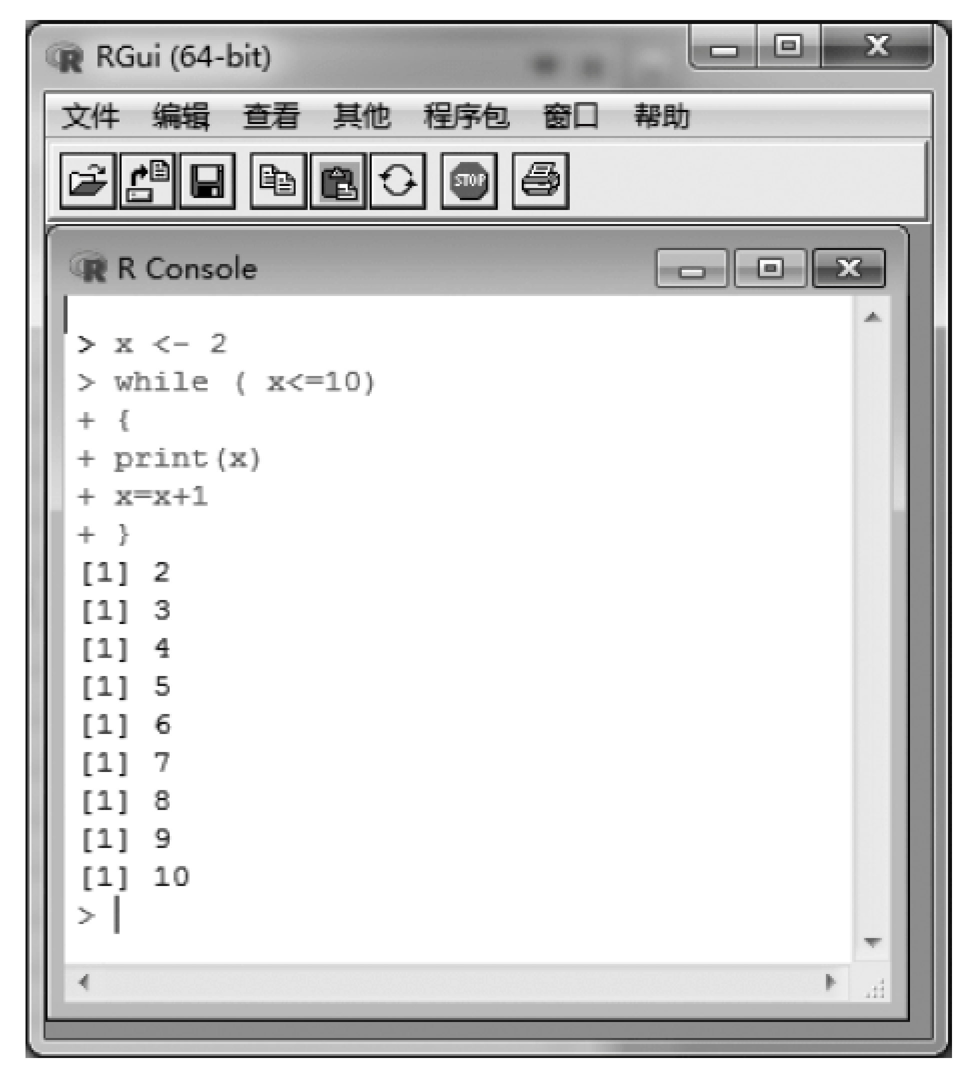


图 8-38 while 语句运行结果

repeat 语句的语法为

```
repeat expr 或 repeat { if(condition) {break}}
```

repeat 是无限循环语句，直到达到循环条件时，使用 break 语句直接跳出循环。

看如下的例子以了解 repeat 语句的用法，通过用户登录次数 userlevel 向量，将用户分为“初级用户”“中级用户”“高级用户”，其中函数 length 用于计算向量中的元素个数，结果如图 8-39 所示。

```
userlevel<-c(1,2,3,11,9,7,18)
```

```
i<-1
results<-""
repeat{
if(i>length(userlevel)) break
if(userlevel [i]<=5) results[i]<-"初级用户" else
if(userlevel [i]<=15) results[i]<-"中级用户" else
results[i]<-"高级用户"
i<-i+1
}
results
```



图 8-39 repeat 语句运行结果

8.5 R语言的数据挖掘和图形绘制包

R 语言在数据挖掘和图形绘制领域也提供了很好的支持，如聚类、分类、关联规则、序列、统计、图表和数据操作等，通过加载不同的 R 语言包就能实现相应的数据挖掘、统计和绘图功能。本节介绍可用于数据挖掘和图形绘制的 R 语言包和函数的集合，包括聚类常用包、分类常用包、关联规则常用包、时间序列常用包、统计常用包和绘图常用包。

1. 聚类包

(1) 常用的包：fpc、cluster、pvclust、mclust。
(2) 基于划分的方法：kmeans、pam、pamk、clara。
(3) 基于层次的方法：hclust、pvclust、agnes、diana。
(4) 基于模型的方法：mclust。

2. 分类包

（1）常用的包：rpart、party、randomForest、rpartordinal、tree、marginTree、maptree、survival。

（2）决策树：rpart、ctree、tree。

（3）随机森林：cforest、randomForest。

（4）回归和 Logistic 回归：glm、predict、residuals。

（5）生存分析：survfit、survdiff、coxph。

（6）支持向量机：e1071。

3. 关联规则包

（1）常用的包：arules。

（2）APRIORI 算法和广度 RST 算法：apriori、drm。

（3）ECLAT 算法、采用等价类、RST 深度搜索和集合的交集：eclat。

4. 时间序列包

（1）常用的包：timsac、arulesSequences。

（2）时间序列构建函数：tseries、forecast。

（3）成分分解：decomp、decompose、stl、tsr。

（4）SPADE 算法：cSPADE。

5. 统计包

（1）常用的包：nlme。

（2）方差分析：aov、anova。

（3）密度分析：density。

（4）假设检验：t. test、prop. test、anova、aov。

（5）线性混合模型：lme。

（6）主成分分析和因子分析：princomp。

6. 绘图包

（1）条形图：barplot。

（2）饼图：pie。

（3）散点图：dotchart。

（4）直方图：hist。

（5）密度图：densityplot。

（6）蜡烛图、箱形图：boxplot。

（7）QQ 图：qqnorm、qqplot、qqline。

（8）双变量图：coplot。

(9) 树图：rpart。

8.6 实际案例

本节以农村危房成因决策树分析为例，对 R 语言进行综合的、实际的案例演示。

```
#设置工作空间
setwd("D:/快盘/2016-2017第一学期/大数据智能分析与处理教材")
#读取数据文件，带标题行
nwdata=read.csv(file="./R语言数据.csv",header=T)

#数据清洗，去掉含缺失值的行
nwdata=na.omit(nwdata)

#为每列数据进行命名
colnames(nwdata) <-c ("nation","population","farmertype","buildyear",
"floor","housearea","roomnumber","buildarea","housetype","buildtype",
"dangerouslevel")

#设置随机数种子
set.seed(0201)
#定义序列 nw，随机抽取 70%的训练集合 30%的测试集
nw <-sample(2,nrow(nwdata),replace=TRUE,prob=c(0.7,0.3))
#把 70%的训练数据赋值给 traindataset
traindataset <-nwdata[nw==1,]
#把 30%的训练数据赋值给 testdataset
testdataset <-nwdata[nw==2,]

#将 dangerouslevel 列转换为 factor 类型
traindataset <-transform(traindataset,dangerouslevel=as.factor
(dangerouslevel))

#安装 CART 决策树包
#install.packages ("tree", dependencies=TRUE)  #如果需要安装，去掉最前面的#号
#安装 ROC 曲线包
#install.packages ("ROCR", dependencies=TRUE)  #如果需要安装，去掉最前面的#号

#加载 CART 决策树包
library(tree)
#加载 ROC 曲线包
library(gplots)
library(ROCR)

#建立决策树模型
```

```
tree.model <- tree(dangerouslevel ~ nation+ population+ farmertype+ buildyear+
floor+ housearea+ roomnumber+ buildarea+ housetype+ buildtype,traindataset)
#概要结论,见图 8-40
summary(tree.model)
```

```
Classification tree:
tree(formula = dangerouslevel ~ nation + population + farmertype +
    buildyear + floor + housearea + roomnumber + buildarea +
    housetype + buildtype, data = traindataset)
Variables actually used in tree construction:
[1] "buildtype"  "farmertype"
Number of terminal nodes:  3
Residual mean deviance:  2.087 = 7176 / 3439
Misclassification error rate: 0.4855 = 1671 / 3442
```

图 8-40　模型概要结论

```
#画决策树图,见图 8-41
plot(tree.model);text(tree.model)
```

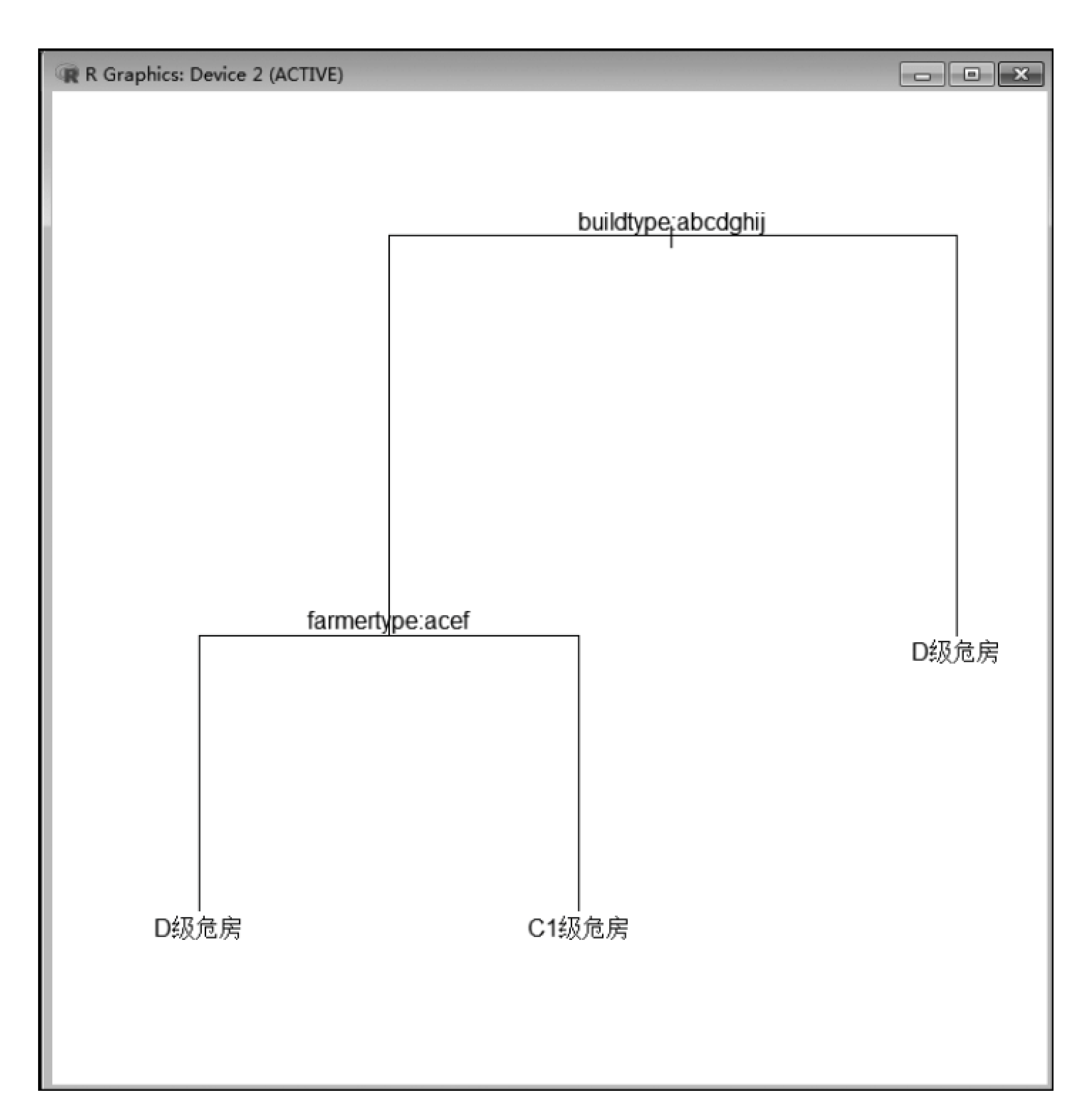


图 8-41　模型决策树图

```
confusion = table (traindataset$ dangerouslevel,predict(tree.model,
traindataset,type = "class"))
accuracy = sum(diag(confusion)) * 100/sum(confusion)

#画 CART 决策的 ROC 曲线,见图 8-42
```

```
tree.pred <- prediction(predict(tree.model,testdataset)[,2],
testdataset$ dangerouslevel)
tree.perf <- performance(tree.pred,"tpr","fpr")
plot(tree.perf)
```

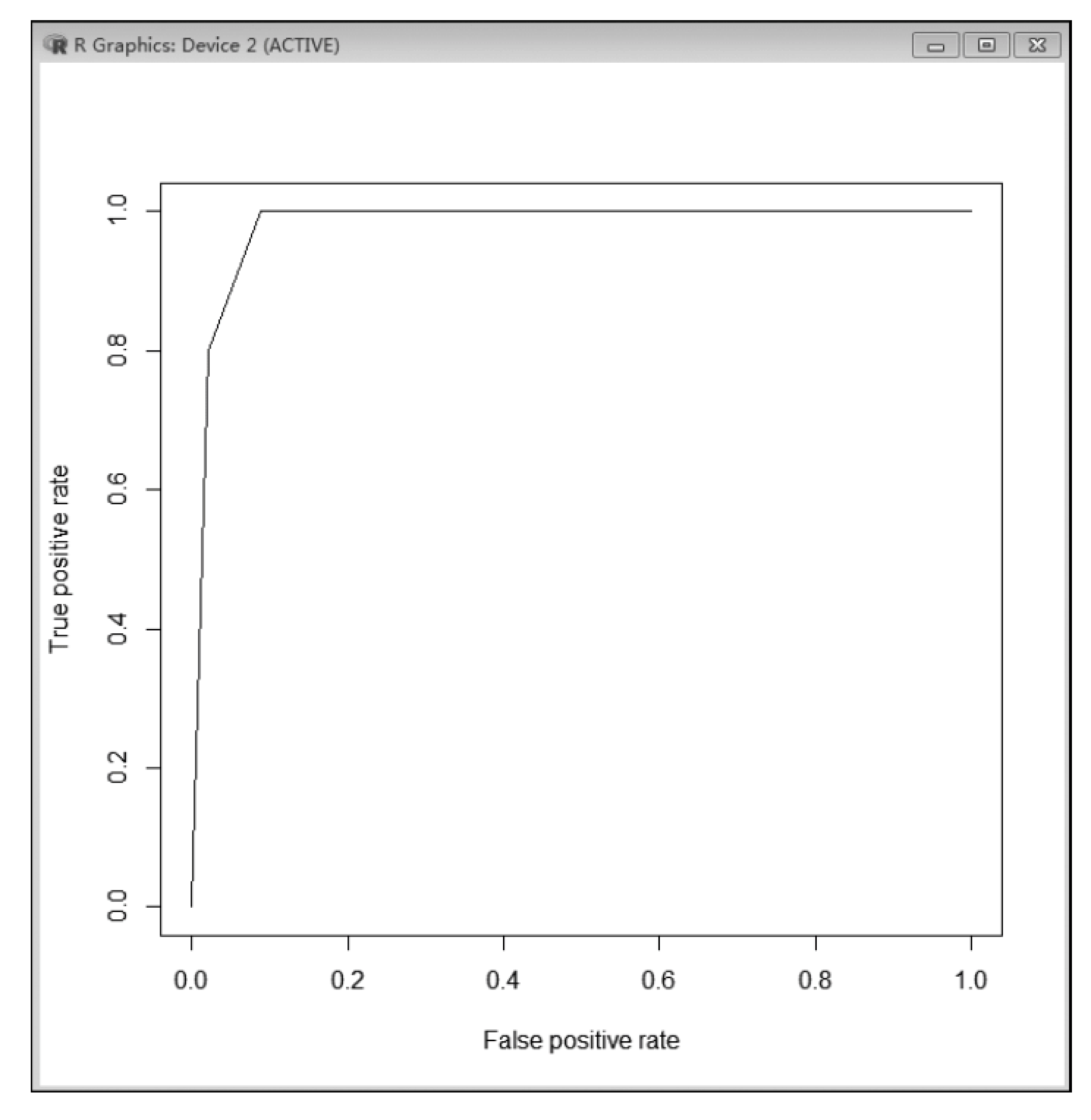


图 8-42　模型 ROC 曲线

8.7　小　结

本章介绍了 R 语言 3.3.1 版本的下载与安装。还介绍了 R 语言的运行、常用操作以及包的使用。接下来介绍了 R 语言的数据结构，包括向量、矩阵、数组、因子、列表框和数据框。还介绍了 R 语言的基本编程结构，包括条件语句和循环语句。在介绍了 R 语言在数据挖掘和图形绘制方面常用的包后，给出一个实际的 R 语言案例。

8.8　习　题

1. 安装聚类算法的 *k*-means 包和随机森林的 randomForest 包。
2. 调用 R 语言随机数函数，产生[50,550]间的均匀随机数 20 个。
3. 利用第 2 题得到的数据，用 plot 语句绘制趋势线图。

8.9 参考文献

[1] 薛毅，陈立萍. R语言实用教程[M]. 北京：清华大学出版社，2014.
[2] 张良均，云伟标，王路，等. R语言数据分析与挖掘实战[M]. 北京：机械工业出版社，2015.
[3] 韩伟. R语言与商业智能[M]. 北京：电子工业出版社，2013.



Hadoop大数据分布式处理生态系统

随着大数据时代的到来，从应用需求、硬件环境、互联模式到计算技术都在发生显著的变化，人们对分布式并行计算的需求也在越来越大，上述变化为分布式并行计算带来了新的发展契机，同时也带来了巨大的研究挑战。其中，多样化的并行计算模型是消除分布式并行应用开发的瓶颈、推动大数据发展的核心技术之一。目前，工业界和学术界正在进行各种尝试和探索，研究和开发不同的并行计算模型，以满足大数据处理的多样化需求。目前有代表性的大数据分布式存储与并行计算的软件框架 Apache Hadoop（以下简称Hadoop）应用非常广泛。时至今日，Hadoop 代表的已经不仅仅是一个分布式数据存储与处理框架，而更多地是指代一个大数据分析处理的生态系统，围绕 Hadoop 这个概念衍生了诸多工具与项目，如 Apache Pic、Apache Hive、Apache HBase、Apache Spark、Apache Storm 等。

本章从介绍 Hadoop 集群的基本概念（9.1 节）开始，然后介绍 HDFS 基本操作（9.2节），MapReduce 并行计算基础（9.3 节），基于 Storm 的分布式实时计算（9.4 节），以及基于 Spark Streaming 的分布式实时计算（9.5 节）。在每一节中都给出若干样例，供读者在实际编程过程中参考。

9.1 Hadoop 集群基础

Hadoop 是一个开源的框架。为实现高可用性，Hadoop 的所有组件在设计之初都是基于如下假设的：集群中硬件故障是常见的，软件框架应当能在软件层面自动处理这些故障。

Hadoop 的基本框架包含 4 个模块：

- Hadoop 公共包（Hadoop Common）。提供其他 Hadoop 模块所需要的库和组件。
- Hadoop 分布式文件系统（Hadoop Distributed File System，HDFS）。提供在普通机器所组成的集群上的高可用的分布式文件系统
- Hadoop YARN。集群计算资源管理平台。
- Hadoop MapReduce。用于大规模数据处理的分布式计算范型。

Hadoop 是典型的主/从（Master/Slave）结构。在数据存储方面，主（Master）节点NameNode 主要负责管理 HDFS 文件系统的命名空间（NameSpace），以及维护文件系统树和文件树中所有文件、文件夹的元数据（metadata）。也就是说 NameNode 知道所有数据及其复本的存放位置、操作日志等。而从（Slave）节点 DataNode 则负责数据块的存取，

是实际存放数据的地方，它们维护并定期向 NameNode 发送自身存储的块(Block)的列表。另外，集群中还有一个 Secondary NameNode 节点，主要功能是定期保存 NameNode 中 HDFS 的元数据的快照，以便在 NameNode 出现问题时接管。

在数据处理方面，主(Master)节点 ResourceManager(RM)负责监控整个集群的可用计算资源，管理运行在 YARN 系统上的分布式应用，而从(Slave)节点 NodeManager (NM)则负责接收 RM 的指令并管理自身所在的单个节点的计算资源。图 9-1 展示了 YARN 框架的基本构成。其中 RM 负责接收作业并将作业分配到 NM。对于某一个作业，在一个 NM 上启动 Application Master(AM)，在其他若干个 NM 上启动 Container。AM 仅负责当前作业的资源调度和请求，而 RM 负责与所有 AM 进行资源的协商与调度。作业实际运行在 Container 上。这样做的好处是：对于不同的应用，作业内部的管理分担到了各个 AM 上，大大减轻了 RM 的压力，提高了集群的可扩展性。

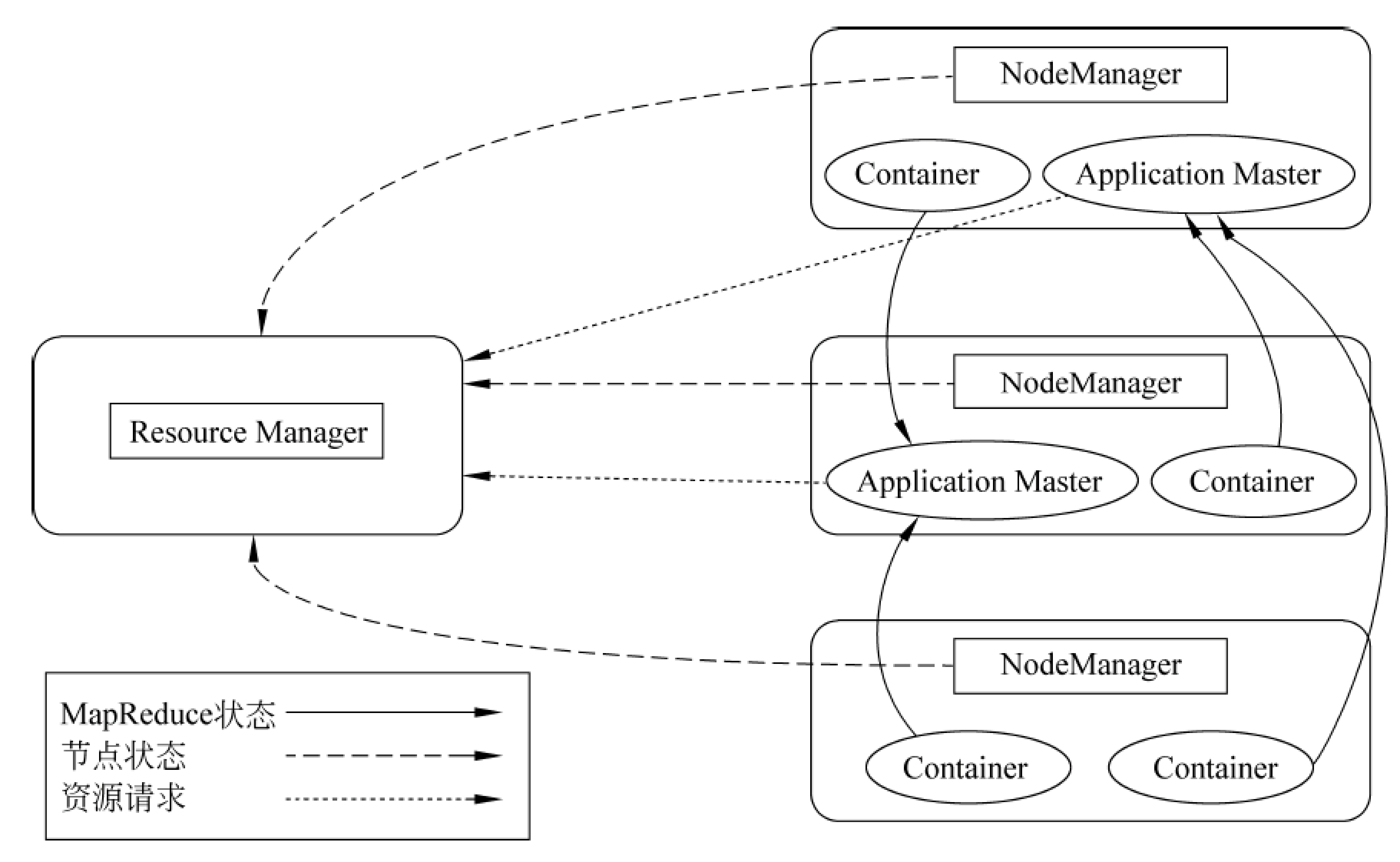


图 9-1 YARN 资源调度框架

综合来看，NameNode、Secondary NameNode 以及 ResourceManager 属于主节点，而 DataNode、NodeManager 属于从节点。一般来说，主节点应当部署在单独的机器上，而不应该和从节点共享硬件。

下面介绍如何安装和配置一个 Hadoop 集群。

9.1.1 Hadoop 安装

Hadoop 可以安装在单台计算机上，也可以安装在一个集群上。这里先来看如何在单台计算机上安装 Hadoop。

1. 安装前的准备

(1) **平台需求**：Hadoop 支持在 GNU/Linux 平台上进行开发和产品化，也支持

Windows 操作系统,但建议使用 Linux 平台。本章所述内容均默认安装在 Linux 平台上。

(2) **软件需求**:目标系统应当安装 Java。Hadoop 2.7 及以上版本的最低需求是 Java 7,Hadoop 2.6 及之前版本支持 Java 6。

另外,目标系统应当安装并运行 ssh。

2. 下载及安装

在 Java 和 ssh 准备就绪之后,到网站 http://www.apache.org/dyn/closer.cgi/hadoop/common/下载一个稳定版本的 Hadoop 并解压到指定的目录。

在运行 Hadoop 之前,需要指定使用的 Java 路径。Hadoop 默认使用系统的 JAVA_HOME 变量,如果用户希望使用该路径,则无须进行任何设置。如果用户希望使用一个新的 Java 路径,则需在解压后目录找到文件 etc/hadoop/hadoop-env.sh 并增加如下变量:

```
# set to the root of your Java installation
export JAVA_HOME=your_java_path
```

保存 etc/hadoop/hadoop-env.sh 并关闭,至此,Java 与 Hadoop 的关联已经建立。下一步是将 Hadoop 的安装目录添加到环境变量,打开~/.bashrc,添加如下语句:

```
# set to the root of your Hadoop installation
export HADOOP_INSTALL = your_hadoop_unpack_path
export PATH=$PATH: $HADOOP_INSTALL/bin: $HADOOP_INSTALL/sbin
```

至此,Hadoop 安装完毕,在终端执行如下命令:

```
$hadoop
```

终端将显示 Hadoop 脚本的使用说明,如图 9-2 所示。

```
Usage: hadoop [--config confdir] [COMMAND | CLASSNAME]
  CLASSNAME            run the class named CLASSNAME
 or
  where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
                       note: please use "yarn jar" to launch
                             YARN applications, not this command.
  checknative [-a|-h]  check native hadoop and compression libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
  classpath            prints the class path needed to get the
  credential           interact with credential providers
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
  trace                view and modify Hadoop tracing settings
```

图 9-2 Hadoop 脚本的使用说明

在终端输入 hadoop version,终端将显示 Hadoop 的版本,如图 9-3 所示。

```
Hadoop 2.7.2
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r b165c4fe8a74265c792ce23f546c64604acf0e41
Compiled by jenkins on 2016-01-26T00:08Z
Compiled with protoc 2.5.0
From source with checksum d0fda26633fa762bff87ec759ebe689c
```

图 9-3 Hadoop 的版本

9.1.2 Hadoop 配置

Hadoop 有 3 种运行模式：本地模式、伪分布式模式以及全分布式模式。前两者都运行在本地计算机上。第 3 种模式运行在真实集群上。

1. 基本配置

Hadoop 有以下 3 种运行模式：

- **本地模式**。在默认情况下，Hadoop 以本地模式运行。本地模式实际上就是 JVM 中的一个进程。此模式常用于 Hadoop 程序的开发与调试。
- **伪分布式模式**。Hadoop 在本地计算机上运行守护进程，模拟一个小规模的集群，用户可以用脚本方式向模拟的集群提交作业。
- **全分布式模式**。Hadoop 在整个集群上运行守护进程。

Hadoop 的配置文件存放在 ect/hadoop/目录中。在最初的版本中，Hadoop 采用一个 hadoop-site.xml 文件来进行配置。从 0.20.0 版本开始，Common、HDFS、MapReduce 和 YARN 分别由 core-site.xml、hdfs-site.xml、mapred-site.xml 和 yarn-site.xml 来进行配置。本地模式无须修改默认配置，伪分布式模式则需做如下修改。

(1) 在配置文件 core-site.xml 中加入以下内容：

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost: 9000</value>
    </property>
</configuration>
```

这个配置项表明 HDFS 文件系统的默认访问路径为 hdfs://localhost:9000。

(2) 在配置文件 hdfs-site.xml 中加入以下内容：

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

这个配置项表明数据只存储一个复本。全分布式模式默认存储 3 个复本，伪分布式模式中，数据存储在本地计算机上，因此没有存储多个复本的必要。

如果要在伪分布式模式中利用 YARN 运行 MapReduce 程序，则还需要进行如下配置。

(1) 在配置文件 mapred-site.xml 中加入以下内容：

```
<configuration>
    <property>
```

```
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

这个配置项表明在此模式下利用 YARN 运行 MapReduce 程序。

（2）在配置文件 yarn-site. xml 中加入以下内容：

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux- services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

至此，伪分布式模式的基本配置结束。在利用伪分布式模式运行 MapReduce 程序前，还需要启动守护进程。

如果是第一次启动，还需要先格式化文件系统：

```
hdfs namenode - format
```

启动 NameNode 和 DataNode 守护进程：

```
start- dfs.sh
```

启动成功后可以在 http://localhost：50070/查看 NameNode 界面，确认 HDFS 系统已经启动，可以进行分布式文件的存取。

如果需要用 YARN 运行 MapReduce 程序，则还需要启动 YARN 守护进程：

```
start- yarn.sh
```

启动成功后可以在 http://localhost：8088/查看资源和节点管理界面。

守护进程所产生的日志文件可以在 Hadoop 安装目录的 logs 文件夹下找到，也可以通过配置 HADOOP_LOG_DIR 来自定义日志文件路径。

2. 集群配置

在集群上安装 Hadoop 的前置条件和在单机上配置 Hadoop 类似，集群中每一台计算机都需要下载 Hadoop 并解压，以及安装对应版本的 Java。限于篇幅，这里仅介绍 Hadoop 非安全模式（non-secure mode）的配置，关于 Hadoop 安全模式的配置，请参考 Hadoop 官方文档（参考文献[3]）。

Hadoop 集群中的每个节点都各自保留一系列的配置文件。Hadoop 支持全局统一的配置，也就是说集群中所有计算机采用同样的配置文件。这种方式对于某些扩展的集群不实用，因为不同批次的计算机的性能很可能不一样。因此通常将计算机划分为不同的计算机类，同一个计算机类的计算机共享配置。实际应用中，建议采用外部工具进行配置管理，如 Chef、Puppet 等。在这里，只讨论通用配置的部分。

在集群安装前，需要对集群中的计算机的功能做一个划分。由于 Hadoop 是主从式结构集群，因此集群中的计算机也有主从之分，并非每台计算机的功能都是相同的。一般来说，NameNode 和 ResourceManager 应当被分开部署在单独的计算机上。有条件的话，Web 应用代理服务器和 MapReduce 作业历史服务器也应当分开部署在单独的计算机上。

Hadoop 的配置文件主要包括两类：一类是各个核心组件，如 HDFS、YARN、MapReduce 等的配置文件，名称格式为 xxx-site. xml，这一类主要负责守护进程的配置；另一类则是环境变量配置脚本，名称格式为 xxx-env. sh，这一类主要负责守护进程的运行环境变量配置。

1）守护进程环境变量配置

用户应当使用 hadoop-env. sh、mapred-env. sh 以及 yarn-env. sh 来对守护进程的环境变量进行配置，表 9-1 是守护进程类型和相应的环境变量部分配置项。

表 9-1　Hadoop 守护进程环境变量配置项

守护进程	配　置　项
NameNode	HADOOP_NAMENODE_OPTS
DataNode	HADOOP_DATANODE_OPTS
Secondary NameNode	HADOOP_SECONDARYNAMENODE_OPTS
ResourceManager	YARN_RESOURCEMANAGER_OPTS
NodeManager	YARN_NODEMANAGER_OPTS
WebAppProxy	YARN_PROXYSERVER_OPTS
MapReduce Job History Server	HADOOP_JOB_HISTORYSERVER_OPTS

比如，如果需要 NameNode 使用并行垃圾回收，则需在 hadoop-env. sh 中进行如下配置：

```
export HADOOP_NAMENODE_OPTS="-XX:+UseParallelGC"
```

2）守护进程配置

（1）HDFS 配置。

HDFS 配置主要指明系统中 NameNode 和 DataNode 的位置、分布式文件块的大小、节点上用于存储分布式文件的本地文件系统路径等。

在 core-site. xml 中，用户需要按表 9-2 进行配置，指明集群中 NameNode 的位置。

表 9-2　NameNode 配置

参　数	值	备　注
fs. defaultFS	NameNode 的路径	hdfs://host: port/

hdfs-site. xml 的配置稍微复杂些，对于 NameNode 和 DataNode 有不同的配置项。

对于 NameNode，常用配置如表 9-3 所示。

表 9-3　HDFS 守护进程环境变量配置项

参　数	值	备　注
dfs. namenode. name. dir	NameNode 在本地文件系统上存储数据的路径	可以用逗号分隔多个路径，这种情况下数据表会在所有路径下都存储一个冗余副本
dfs. hosts /dfs. hosts. exclude	允许/拒绝的 DataNode	用于控制允许/拒绝 DataNode
dfs. blocksize	268435456	块大小（字节），大文件系统可以采用 256MB
dfs. namenode. handler. count	100	用于与 DataNode 进行远程通信的句柄数

对于 DataNode，常用的配置如表 9-4 所示。

表 9-4　DataNode 配置项

参　数	值	备　注
dfs. datanode. data. dir	DataNode 在本地文件系统中存储数据块的文件夹的路径	可以用逗号分隔多个路径

（2）YARN 配置

YARN 配置主要指明集群中 ResourceManager 和 NodeManager 的位置、ResourceManager 与从节点进行各项调度与交互的通信接口、单个任务的可用资源、NodeManager 用于运行分布式程序的可用资源等。这些配置集中在 yarn-site. xml。

如果要在集群中使用 YARN 对 MapReduce 进行调度，则还需要对 yarn-site. xml 进行配置。ResourceManager 和 NodeManager 都共用的部分配置项如表 9-5 所示。

表 9-5　YARN 共用配置项

参　数	值	备　注
yarn. acl. enable	true/false	是否启用 ACLs，默认为 false
yarn. admin. acl	Admin ACL	管理员列表，以逗号分隔，默认值为 *，代表所有人都具有管理员权限，空格表示无人具有管理员权限
yarn. log-aggregation-enable	true/false	启用或禁用日志聚合，默认为 false

针对 ResourceManager 的部分配置项如表 9-6 所示。

表 9-6　ResourceManager 配置项

参　数	值	备　注
yarn. resourcemanager. address	host：port	用于 YARN 客户端提交作业的主机与端口地址，如设置，将覆盖 yarn. resourcemanager. hostname 的值
yarn. resourcemanager. scheduler. address	host：port	用于 AM 向 RM 请求资源的主机与端口地址，如设置，将覆盖 yarn. resourcemanager. hostname 的值
yarn. resourcemanager. resource-tracker. address	host：port	用于 NM 与 RM 进行通信的主机与端口地址，如设置，将覆盖 yarn. resourcemanager. hostname 的值

续表

参　　数	值	备　　注
yarn. resourcemanager. admin. address	host：port	用于管理员向 RM 发送管理命令的主机与端口地址，如设置，将覆盖 yarn. resourcemanager. hostname 的值
yarn. resourcemanager. webapp. address	host：port	RM 的 Web UI 地址，如设置，将覆盖 yarn. resourcemanager. hostname 的值
yarn. resourcemanager. hostname	host.	设置单个 RM 的 host 地址时，将同时自动以默认端口设置所有其他地址(但会被相应的设置覆盖)
yarn. resourcemanager. scheduler. class	Scheduler class	调度器主类，可选 CapacityScheduler、FairScheduler 以及 FifoScheduler
yarn. scheduler. minimum-allocation-mb	1024	单个 Container 可分配的最小内存，以兆字节(MB)为单位
yarn. scheduler. maximum-allocation-mb	8192	单个 Container 可分配的最大内存，以兆字节(MB)为单位

针对 NodeManager 的部分配置项如表 9-7 所示。

表 9-7　NodeManager 配置项

参　　数	值	备　　注
yarn. nodemanager. resource. memory-mb	8192	NodeManager 上可用于执行 YARN 任务的内存数，以兆字节(MB)为单位
yarn. nodemanager. vmem-pmem-ratio	2	单个任务可用的虚拟内存与物理内存的比值
yarn. nodemanager. local-dirs	path1，path2，…	本地文件系统上用于存储中间结果的文件夹路径，以逗号分隔
yarn. nodemanager. log-dirs	path1，path2，…	本地文件系统上用于存储日志的文件夹路径，以逗号分隔
yarn. nodemanager. aux-services	mapreduce_shuffle	用于 MapReduce 应用的 shuffle 服务设置

(3) MapReduce 配置。

MapReduce 的配置集中在 maprd-site. xml，主要负责运行框架的选择、Map 与 Reduce 及其子线程的资源限制等。

针对 MapReduce 应用的常用配置项如表 9-8 所示。

表 9-8　MapReduce 配置项

参　　数	值	备　　注
mapreduce. framework. name	yarn	运行框架配置
mapreduce. map. memory. mb	1536	Map 的内存限额，以兆字节(MB)为单位
mapreduce. map. java. opts	-Xmx1024M	Map 的子线程的 JVM 最大可用内存
mapreduce. reduce. memory. mb	3072	Reduce 的内存限额，以兆字节(MB)为单位
mapreduce. reduce. java. opts	-Xmx2560M	Reduce 的子线程的 JVM 最大可用内存

续表

参　　数	值	备　　注
mapreduce. task. io. sort. mb	512	用于提高效率进行数据排序的内存限额，以兆字节(MB)为单位
mapreduce. task. io. sort. factor	100	合并数据流时一次合并的数据流数目，每次合并时选择最小的前 100 个进行合并
mapreduce. reduce. shuffle. parallelcopies	50	shuffle 阶段 Reduce 从 Map 获取数据时并行传输数据的复本数

针对 JobHistory 服务器的常用配置如表 9-9 所示。

表 9-9　JobHistory 配置项

参　　数	值	备　　注
mapreduce. jobhistory. address	host：port	JobHistory 服务器地址，默认端口 10020
mapreduce. jobhistory. webapp. address	MapReduce JobHistory Server Web UI host：port	JobHistory Web UI 地址，默认端口 19888
mapreduce. jobhistory. intermediate-done-dir	/mr-history/tmp	MapReduce 作业产生的历史文件的存放路径
mapreduce. jobhistory. done-dir	/mr-history/done	MR JobHistory Server 管理的历史文件的存放路径

9.2　HDFS 基础操作

在 Hadoop 集群中，用户可以采用文件系统命令 FS shell(File System shell)来访问 HDFS 文件系统中的文件。FS shell 不仅可以用来与 HDFS 进行交互，也可以与 Hadoop 支持的其他文件系统进行交互，如本地文件系统、HFTP 文件系统、S3 文件系统等。FS shell 通过如下方式进行调用：

```
hadoop fs <args>
```

所有 FS shell 命令都以路径 URL 作为参数，路径 URL 的格式为 scheme://authority/path。对于 HDFS 文件系统而言，scheme 是 hdfs；对于本地文件系统而言，scheme 则是 file。scheme 和 authority 都是可选的，在没有特别指定的情况下，将默认为配置文件中的 scheme。因此，一个 HDFS 文件或文件夹 hdfs://namenodehost/parent/child 可以简写为 /parent/child(前提是配置文件中将 hdfs://namenodehost 设置为默认文件路径)。

下面介绍 HDFS 常用操作命令。

1. appendToFile

用法：hadoop fs -appendToFile <localsrc> … <dst>

功能：将一个或多个本地文件追加到目标文件。也可以从标准输入追加到目标文件。

用例：hadoop fs -appendToFile localfile /user/hadoop/hadoopfile
hadoop fs -appendToFile localfile1 localfile2 /user/hadoop/hadoopfile
hadoop fs -appendToFile localfile hdfs://nn.example.com/hadoop/hadoopfile
hadoop fs -appendToFile - hdfs://nn.example.com/hadoop/hadoopfile

返回值：成功返回0,出现错误返回－1。

2. cat

用法：hadoop fs -cat URI [URI …]

功能：将目标文件复制到标准输出。

用例：hadoop fs -cat hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2
hadoop fs -cat file:///file3 /user/hadoop/file4

返回值：成功返回0,出现错误返回－1。

3. checksum

用法：hadoop fs -checksum URI

功能：计算目标文件的总校验和。

用例：hadoop fs -checksum hdfs://nn1.example.com/file1
hadoop fs -checksum file:///etc/hosts

4. chgrp

用法：hadoop fs -chgrp [-R] GROUP URI [URI …]

功能：更改目标文件的group。操作者必须是文件的所有者或超级用户。

选项：-R　对文件夹内的文件递归地执行此命令。

5. chmod

用法：hadoop fs -chmod [-R] <MODE[,MODE]… | OCTALMODE> URI [URI …]

功能：更改目标文件的权限。操作者必须是此文件的所有者或者超级用户。

选项：-R　对文件夹内的文件递归地执行此命令。

6. chown

用法：hadoop fs -chown [-R] [OWNER][:[GROUP]] URI [URI]

功能：更改目标文件的所有者。操作者必须是超级用户。

选项：-R　对文件夹内的文件递归地执行此命令。

7. copyFromLocal

用法：hadoop fs -copyFromLocal <localsrc> URI

功能：与 put 命令相似，但是 copyFromLocal 只能复制本地文件到 HDFS 中。

选项：如果它已经存在，-f 会覆盖目标。

8. copyToLocal

用法：hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>

功能：把文件从 HDFS 上下载到本地。除了限定目标路径是一个本地文件，其他与 get 命令相似。

9. count

用法：hadoop fs -count [-q] [-h] [-v] <paths>

功能：匹配指定的模式，计算目录、文件和字节的数量。输出的列分别是当前路径下的文件夹个数、当前文件夹下文件的个数、该文件夹下文件所占的空间大小、当前路径。

选项：-q　输出的列是限额、剩余限额、空间限额、剩余空间限额。

　　-h　将文件大小的数值用方便阅读的形式表示，比如用 64.0M 代替 67108864。

　　-v　显示一个标题行。

用例：hadoop fs -count hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2

　　hadoop fs -count -q hdfs://nn1.example.com/file1

　　hadoop fs -count -q -h hdfs://nn1.example.com/file1

　　hdfs dfs -count -q -h -v hdfs://nn1.example.com/file1

返回值：成功返回 0，出现错误返回 −1。

10. cp

用法：hadoop fs -cp [-f] [-p | -p[topax]] URI [URI …] <dest>

功能：将文件复制到目标路径，这个命令允许同时复制多个文件，但是在这种情况下目标路径必须是一个目录。

如果下列要求被满足，则在复制时可以保留"raw.*"命名空间扩展属性：

(1) 源文件系统和目标文件系统支持(仅限 HDFS)。

(2) 所有源路径名和目标路径名都在/.reserved/raw 层次结构下。

选项：-f　当文件存在时，进行覆盖。

　　-p　将权限、所属组、时间戳、ACL 以及 xattr 等也进行复制。

用例：hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2

　　hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir

返回值：成功返回 0，出现错误返回 −1。

11. df

用法：hadoop fs -df [-h] URI [URI …]

功能：显示剩余空间。

选项：-h　将文件大小的数值用方便阅读的形式表示，比如用 64.0M 代替 67108864。

用例：hadoop dfs -df /user/hadoop/dir1

12. du

用法：hadoop fs -du [-s] [-h] URI [URI…]

功能：如果参数为目录，显示该目录下所有目录和文件的大小；如果参数为单个文件，则显示文件大小。

选项：-s　指输出所有文件大小的累加和，而不是每个文件的大小。

　　-h　会将文件大小的数值用方便阅读的形式表示，比如用 64.0M 代替 67108864。

用例：hadoop fs -du /user/hadoop/dir1 /user/hadoop/file1

返回值：成功返回 0，出现错误返回 −1。

13. find

用法：hadoop fs -find <path> … <expression> …

功能：查找满足表达式的文件和文件夹。如果 path 没有配置，默认的就是当前目录，如果 expression 没有配置，则默认为-print。

选项：-name pattern　如果文件名匹配 pattern 则处理，不区分大小写。

　　-iname pattern　如果文件名匹配 pattern 则处理，大小写敏感。

　　-print　将当前路径名写入标准输出。

　　-print0　将当前路径名写入到标准输出，并追加一个 ASCII 空字符。

用例：hadoop fs -find / -name test -print

返回值：成功返回 0，出现错误返回 −1。

14. get

用法：hadoop fs -get [-ignorecrc] [-crc] <src> <localdst>

功能：将文件复制到本地文件系统。CRC 校验失败的文件可通过-ignorecrc 选项复制。文件和 CRC 校验和可通过-crc 选项一起复制。

用例：hadoop fs -get /user/hadoop/file localfile

　　hadoop fs -get hdfs://nn.example.com/user/hadoop/file localfile

返回值：成功返回 0，出现错误返回 −1。

15. help

用法：hadoop fs -help

功能：返回使用输出。

16. ls

用法：hadoop fs -ls [-d] [-h] [-R] [-t] [-S] [-r] [-u] <args>

功能：对于一个文件，该命令返回的内容以如下格式列出：文件权限，复本个数，用户 ID，组 ID，文件大小，最近一次修改日期，最近一次修改时间，文件名。

对于一个目录，该命令返回这一目录下的第一层子目录和文件，与 UNIX 中 ls 命令的结果类似，返回的内容以如下格式列出：文件权限，用户 ID，组 ID，最近一次修改日期，最近一次修改时间，文件名。一个目录的文件通过默认的文件名排序。

选项：-d　以纯文件的形式展示目录。

-h　将文件大小的数值用方便阅读的形式表示，比如用 64.0M 代替 67108864。

-R　递归地列出子目录。

-t　修改时间排序输出（最近的）。

-S　文件大小排序输出。

-r　倒序排序。

-u　使用访问时间而不是修改时间显示和排序。

用例：hadoop fs -ls /user/hadoop/file1

返回值：成功返回 0，出现错误返回－1。

17. mkdir

用法：hadoop fs -mkdir [-p] <paths>

功能：以<paths> 中的 URI 作为参数，创建目录。

选项：-p 与 UNIX 中 mkdir -p 的用法相似。这一路径上的父目录如果不存在，则创建父目录。

用例：hadoop fs -mkdir /user/hadoop/dir1 /user/hadoop/dir2

hadoop fs -mkdir hdfs://nn1.sample.com/dir hdfs://nn2. sample.com/dir

返回：成功返回 0，出现错误返回－1。

18. moveFromLocal

用法：hadoop fs -moveFromLocal <localsrc> <dst>

功能：与 put 命令类似，但是本地源文件复制之后自身会被删除。

19. moveToLocal

用法：hadoop fs -moveToLocal [-crc] <src> <dst>

功能：暂未实现此命令。

20. mv

用法：hadoop fs -mv URI [URI …] <dest>

功能：将文件从源路径移动到目标路径（移动之后源文件被删除）。目标路径为目录的情况下，源路径可以有多个。跨文件系统的移动（本地到 HDFS 或者反过来）是不允许的。

用例：hadoop fs -mv /user/hadoop/file1 /user/hadoop/file2

hadoop fs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2

hdfs://nn.example.com/file3 hdfs://nn.example.com/dir1

返回值：成功返回 0，出现错误返回 -1。

21. put

用法：hadoop fs -put <localsrc> … <dst>

功能：将单个的源文件 src 或者多个源文件 srcs 从本地文件系统复制到目标文件系统中（<dst> 对应的路径）。也可以从标准输入中读取输入并写入目标文件系统中。

用例：hadoop fs -put localfile /user/hadoop/hadoopfile

hadoop fs -put localfile1 localfile2 /user/hadoop/hadoopdir

hadoop fs -put localfile hdfs://nn.example.com/hadoop/hadoopfile

hadoop fs -put - hdfs://nn.example.com/hadoop/hadoopfile

返回值：成功返回 0，出现错误返回 -1。

22. rm

用法：hadoop fs -rm [-f] [-r |-R] [-skipTrash] URI [URI …]

功能：删除参数指定的文件。

选项：-f 如果文件不存在，则不显示诊断信息和修改退出状态来反映错误。

-R 递归地删除目录和目录下的内容。

-r 和-R 的功能一样。

-skipTrash 会绕过回收站，如果启用，会立即删除指定的一个或多个文件，当需要删除来自超额目录的文件时可以使用这个选项。

用例：hadoop fs -rm hdfs://nn.example.com/file /user/hadoop/emptydir

返回值：成功返回 0，出现错误返回 -1。

23. rmdir

用法：hadoop fs -rmdir [--ignore-fail-on-non-empty] URI [URI …]

功能：删除目录。

选项：--ignore-fail-on-non-empty 如果目录还包含文件，使用通配符不会失败。

用例：hadoop fs -rmdir /user/hadoop/emptydir

24. setrep

用法：hadoop fs -setrep [-R] [-w] <numReplicas> <path>

功能：改变一个文件的复本个数。如果路径是一个目录，这个命令递归地改变在这个根目录下所有文件的复本个数。

选项：-w 请求命令等待复制完成，这可能需要花费很长的时间。

-R 可以对一个目录下的所有目录和文件递归地执行改变复本个数的操作。

用例：hadoop fs -setrep -w 3 /user/hadoop/dir1

返回值：成功返回 0，出现错误返回－1。

25. tail

用法：hadoop fs -tail [-f] URI

功能：显示最后 1kb 的内容。

选项：-f 的用法与 UNIX 类似，也就是说当文件尾部添加了新的数据或者做出了修改时，在标准输出中也会刷新显示。

用例：hadoop fs -tail pathname

返回值：成功返回 0，出现错误返回－1。

26. text

用法：hadoop fs -text <src>

功能：将文本文件或者某些格式的非文本文件通过文本格式输出。允许的格式有 zip 和 TextRecordInputStream。

27. touchz

用法：hadoop fs -touchz URI [URI …]

功能：创建一个大小为 0 的文件。

用例：hadoop fs -touchz pathname

返回值：成功返回 0，出现错误返回－1。

28. truncate

用法：hadoop fs -truncate [-w] <length> <paths>

功能：将文件按照 length 进行截取，可以理解成截取[1/length]部分。

选项：-w　请求命令等待块恢复完成。没有-w 标志时，当恢复正在进行的时候，这个文件可能保持非关闭状态一段时间。在此期间文件不能再打开追加内容。

用例：hadoop fs -truncate 55 /user/hadoop/file1 /user/hadoop/file2

hadoop fs -truncate -w 127 hdfs://nn1.example.com/user/hadoop/file1

29. usage

用法：hadoop fs -usage command

功能：返回命令的 help 信息。

HDFS 文件系统提供了大数据分析的分布式数据存储与访问功能。以 HDFS 为基础，MapReduce 提供了一种分布式数据处理编程范式，YARN 提供了分布式计算任务的资源调度功能。三者有机结合，组成了 Hadoop 大数据分析处理框架。

9.3 MapReduce并行计算框架

对相互间不具有计算依赖关系的大数据，实现并行最自然的办法就是采取分而治之的策略。MapReduce 的核心思想是：将处理过程中的两个主要处理阶段提炼为一种抽象的操作机制，借鉴函数式程序设计语言 Lisp 中的思想，定义了 Map 和 Reduce 两个抽象的操作函数。Map 阶段主要负责数据的处理并转化成中间结果，Reduce 阶段主要负责收集中间结果并计算输出。Map 和 Reduce 函数在算法处理中承担的任务角色如图 9-4 所示。

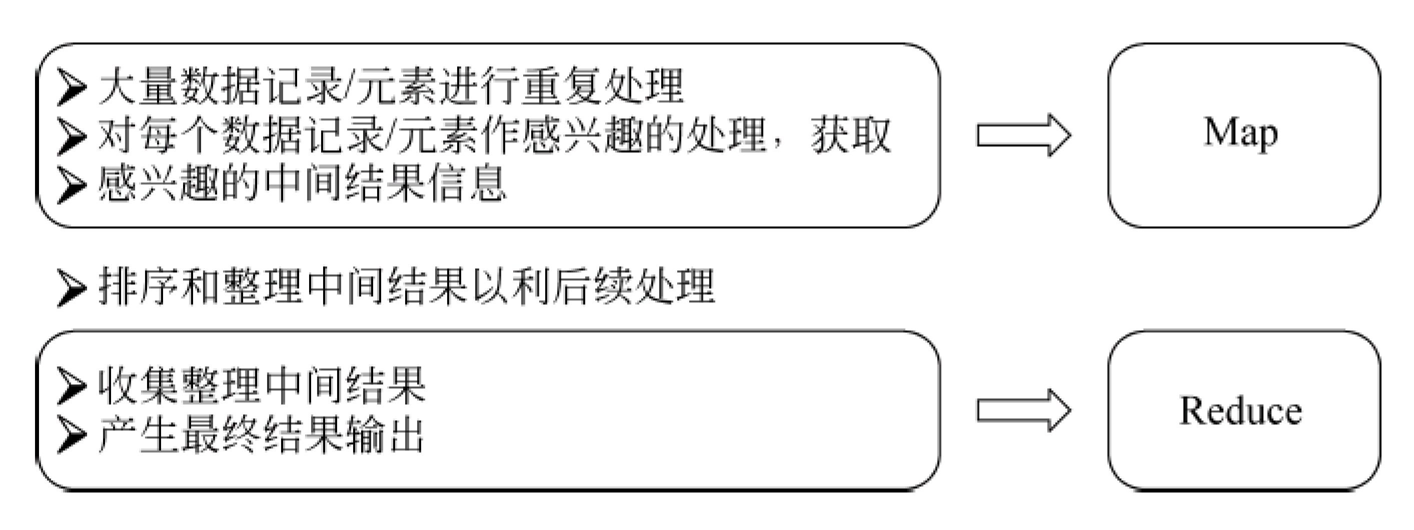


图 9-4 Map 和 Reduce 函数在算法处理中承担的任务角色

- 上升到抽象模型：Mapper 与 Reducer。MPI 等并行计算方法缺少高层并行编程模型，为了克服这一缺陷，MapReduce 借鉴了 Lisp 函数式语言中的思想，用 Map 和 Reduce 两个函数提供了高层的并行编程抽象模型。
- 上升到构架：统一构架，为程序员隐藏系统层细节。MPI 等并行计算方法缺少统一的计算框架支持，程序员需要考虑数据存储、划分、分发、结果收集、错误恢复等诸多细节。为此，MapReduce 设计并提供了统一的计算框架，为程序员隐藏了绝大多数系统层面的处理细节。

下面从逻辑实体的角度回顾下 MapReduce 的运行机制，按照时间顺序包括输入分片(input split)阶段、Map 阶段、Combiner 阶段、Shuffle 阶段和 Reduce 阶段。

(1) **输入分片(input split)**阶段。在进行 Map 计算之前，MapReduce 会根据输入文件计算输入分片，每个输入分片针对一个 Map 任务，输入分片存储的并非数据本身，而是一个分片长度和一个记录数据的位置的数组，假设有 3 个输入文件 A、B、C，大小分别是 7MB、67MB 和 127MB，那么 MapReduce 会把文件 A 分为一个输入分片，文件 B 则是两个输入分片，而文件 C 也是两个输入分片。显而易见，如果在 Map 计算前做输入分片调整(合并小文件)，那么就会有 5 个 Map 任务执行并处理大小不均的数据。

(2) **Map 阶段**。由程序员编写 Map 函数，相对好控制，一般在数据存储节点上做本地化操作。

(3) **Combiner 阶段**。这个阶段是程序员可以选择的，是一个本地化的 Reduce 操作。作为 Map 的后续操作，主要是在 Map 计算出中间文件前做一个简单的合并重复 key 值的操作。在 Reduce 操作前对相同的 key 做合并操作会大幅减少文件大小，提高传输效率。

(4) **Shuffle 阶段**。是将 Map 的输出作为 Reduce 的输入的过程，也是 MapReduce 算法需要重点进行优化的地方。

(5) **Reduce 阶段**。由程序员编写 Map 函数，计算最终结果并输出到 HDFS 上。

MapReduce 是 Hadoop 体系中用于编写大数据并行程序的软件框架。在这个框架中，用户可以专注于程序本身的功能，而不用在如何将程序并行化上花费太多精力。通常来说，一个 MapReduce 作业会利用 Map 将输入数据划分为独立的块并完全独立地处理。在 Map 处理完数据之后，框架将 Map 的输出排序，并作为 Reduce 的输入。一般来说输入和输出都存放在同一个文件系统中。

MapReduce 通过键值对<key，value>的方式来进行计算。也就是说，在 MapReduce 中，输入数据全部被当作<key，value>的格式进行处理，输出也是<key，value>的格式。key 和 value 的类都实现了 Writeable 接口，因此都是可序列化的，这样才能将中间结果和输出写到文件系统中。另外，由于 key 需要进行比较与排序，因此需要实现 WritableComparable 接口。MapReduce 自带若干 key 和 value 的类，都实现了相应的接口。

9.3.1 MapReduce 程序实例：WordCount

WordCount 是一个单词计数程序，功能是计算指定文件夹内的文件内容中不同单词出现的次数。

程序代码如下：

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable>{
        private final static IntWritable one=new IntWritable(1);
        private Text word=new Text();

        public void map (Object key, Text value, Context context
                        ) throws IOException, InterruptedException {
```

```
            StringTokenizer itr=new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable>{
        private IntWritable result=new IntWritable();
        public void reduce(Text key, Iterable<IntWritable>values, Context
                   context) throws IOException, InterruptedException{
            int sum=0;
            for (IntWritable val : values) {
                sum +=val.get(); }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf=new Configuration();
        Job job=Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true)? 0:1);
    }
}
```

将此程序打包成 wc.jar，那么就可以通过如下脚本来运行 WordCount 程序：

```
hadoop jar wc.jar WordCount inputDirURL outputDirURL
```

其中 wc.jar 代表要运行的程序，WordCount 指定了入口类，inputDirURL 指定了需要进行词数统计的文件所在的文件夹路径，outputDirURL 指定了统计结果的输出路径。

9.3.2 Hadoop Streaming

Hadoop Streaming 是标准 Hadoop 发行版的一个组件，可以让用户指定任意的可执

行程序或脚本作为 Mapper 或 Reducer 来执行 MapReduce 程序。一个显而易见的好处是，通过 Hadoop Streaming，用户的变成语言将不再局限于 Java，而可以使用 Python、Linux shell 等一系列语言和工具。

在 Hadoop Streaming 中，数据以标准输入/标准输出的形式进行交互。Mapper 从标准输入接收要处理的数据，经过处理后的中间结果逐行输出到标准输出；Reducer 从标准输入逐行接收中间结果，进行汇总后再逐行输出到标准输出。mapper 的初始化、数据的分发等由 Hadoop Streaming 来完成。

Hadoop Streaming 的使用实例如下：

```
hadoop jar hadoop-streaming-2.7.0.jar \
  -input myInputDirs \
  -output myOutputDir \
  -mapper myMapper.py \
  -reducer myReducer.py
```

这个实例的功能读取 myInputDirs 文件夹下文件的内容，逐行输出到标准输出，并调用 myMapper. py 对这些内容进行处理，接着调用 myReducer. py 处理中间结果，然后接收汇总结果最终输出到 myOutputDir。这里的 myMapper. py 与 myReducer. py 是用户根据自己的应用需求编写的 Python 程序。

9.4 基于 Storm 的分布式实时计算

9.4.1 Storm 简介

Strom 是 Apache 基金下的一个免费、开源分布式实时计算项目。与其他实时大数据分析处理技术不同，Storm 可以进行流式数据的实时分析预处理。Storm 可以用于实时分析、在线机器学习、持续计算、分布式远程过程调用(Remote Procedure Call，RPC)、数据提取-转换-加载(Extraction-Transformation-Loading)等。Storm 具有在单个节点上每秒处理百万级数据元组的能力，同时它具有高度可扩展性、容错性，因此非常易于使用和部署。

9.4.2 Storm 基本概念

一个 Storm 应用程序对应的分布式计算结构称为 Topology，这也是 Strom 中最重要的概念。一个 topology 由 Spout(数据发生器)、Bolt(数据计算单元)组成。数据以流的方式从数据源进入 Spout，并转换为多个 Tuple 组成的 stream，分发到不同的 Bolt 进下计算。一个典型的 Topology 如图 9-5 所示。

由于 Spout 的存在，Storm 可以处理不同的数据流来源。实际上，将数据流转换为 Tuple 流的工作就在 Spout 中完成，而且主要依靠代码编写者来完成。Storm 可以处理的数据源包括：

• Web 日志。

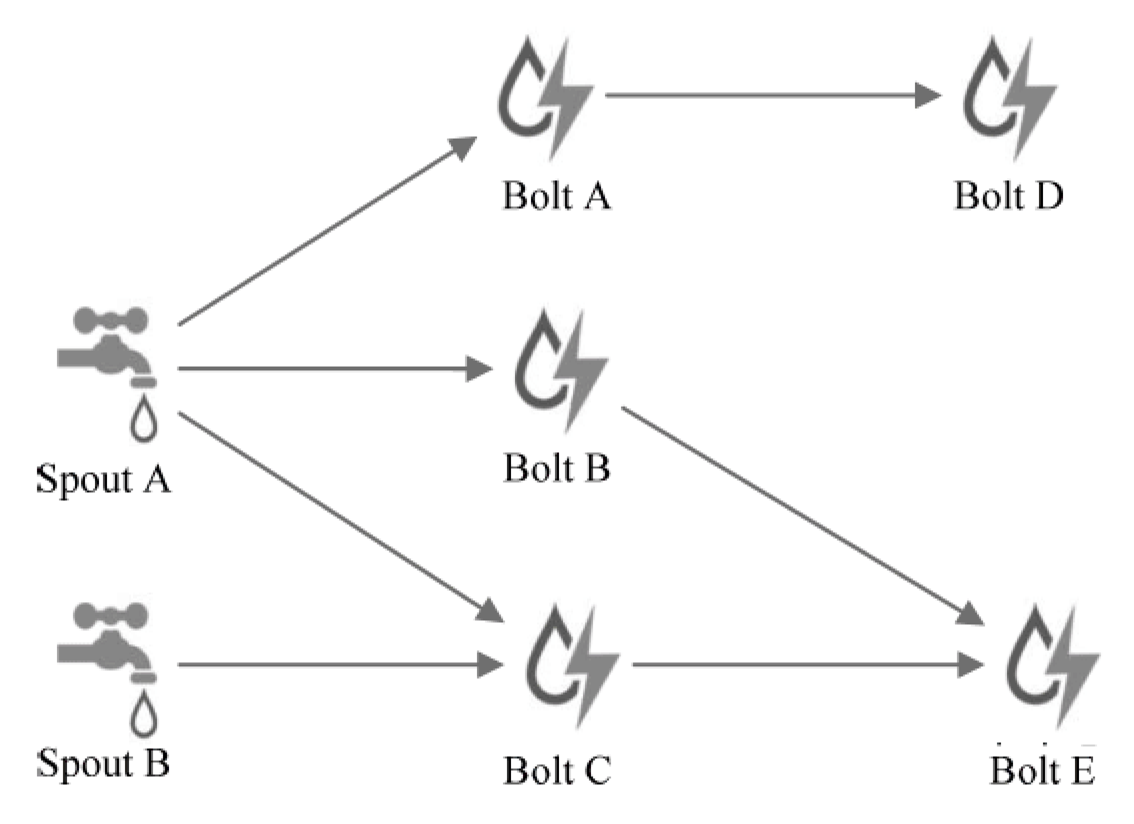


图 9-5 Strom 中 Topology 结构示意图

- 社交网络消息流。
- 传感器的实时数据流。

当然，只要能在 Spout 中编写合适的转换代码，Storm 也可以其他类型的流数据。在设计 Storm 程序时，不建议在 Spout 中添加数据计算的功能，而建议 Spout 只负责转换，计算代码全部放到 Bolt 中去。

Bolt 的功能则是订阅(subscribe)一个或多个 Tuple 流，执行计算，然后输出一个或多个 Tuple 流。通过订阅一个或多个 Spout/Bolt 发射的 Tuple 流，Storm 可以构建非常复杂的 Tuple 流网络。如在图 9-5 中，Bolt C 接收 Spout A 和 Spout B 发出的 Tuple 流，执行计算完毕后转换为新的 Tuple 流并发射到 Bolt E。

Bolt 可以执行的功能包括：

- 数据过滤。
- 函数计算。
- 数据聚合(aggregation)与连接(join)。
- 数据库读写等。

在 Storm 中，数据流 Stream 不停地从 Spout 产生并发射到 Bolt 运行。虽然同一个数据是有序地从 Spout 按照 topology 的拓扑结构流经各个 Bolt 的，但在同一时刻，Spout 与 Bolt 可以并行地处理不同的数据。Storm 的并行不仅体现在 Spout 与 Bolt 的拓扑结构中，同样体现在 Spout 与 Bolt 的内部。每一个 Spout 或 Bolt 都可以利用一个或多个任务(Task)来完成，而 Task 是并行地执行在集群的工作节点上的。图 9-6 展示了 Spout、Bolt 与 Task 之间的示意图，其中 Spout 由两个 Task 实现，Bolt A、B、C 分别由 4、3、2 个 Task 实现，整个 Topology 使用了 11 个 Task(在图中以圆圈表示)。每一个 Task 在实际的机器上由一个线程(thread)来运行。

1. Storm 集群架构

Storm 优秀的实时处理性能依赖于其专门针对流数据处理而设计的架构。一个 Topology 可能包含若干个 Task，那么，这些 Task 是如何在一个集群上协同完成一个分

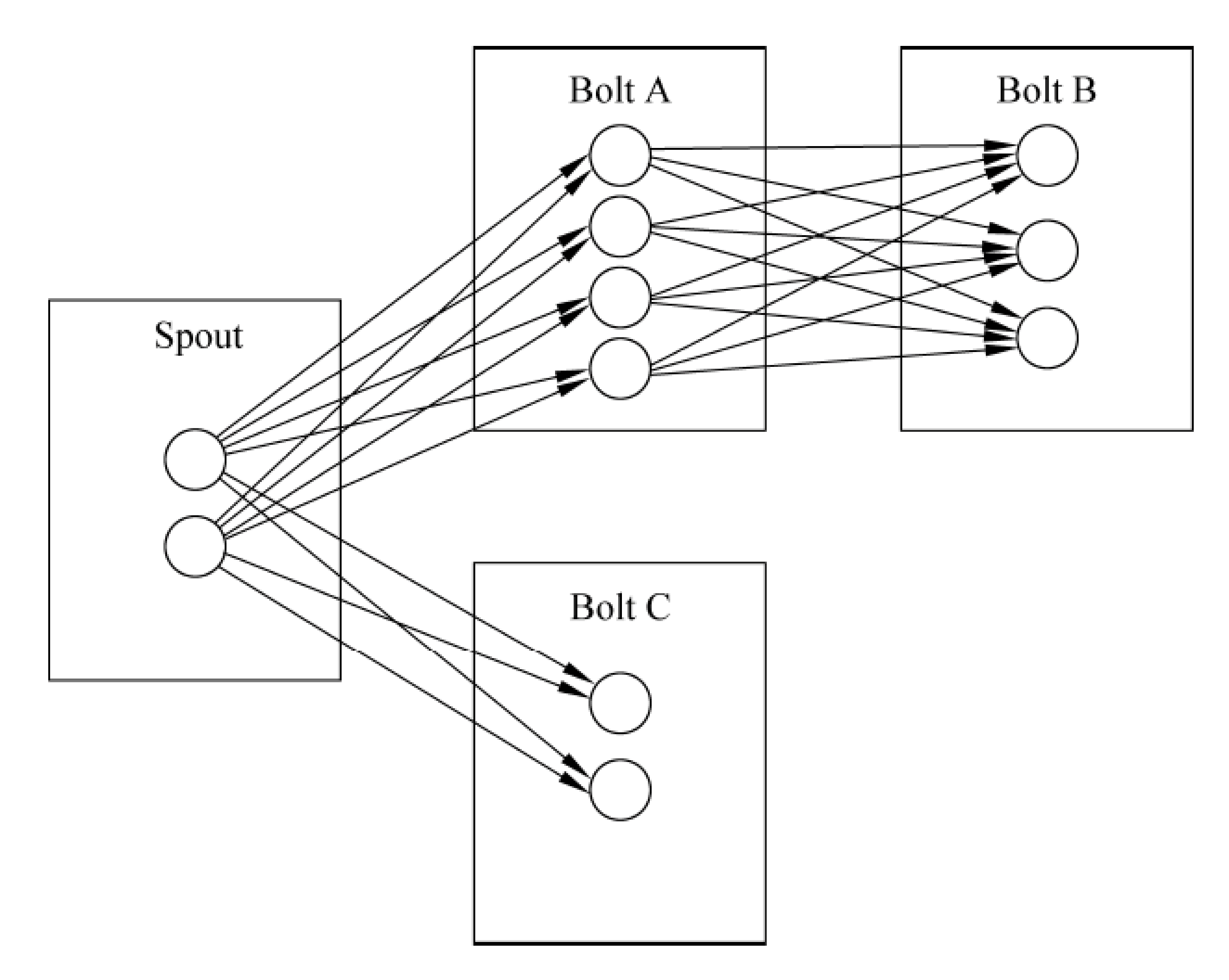


图 9-6　Spout、Bolt 与 Task 的关系示意图

布式实时计算任务的呢？下面看一下 Storm 的集群架构。

图 9-7 展示了 Storm 的基本集群架构。其中主节点（Master Node）与工作节点（Worker Node）为实际的集群中的机器。在主节点上，运行守护进程 Nimbus，在工作节点上，运行守护进程 Supervisor。Nimbus 的主要功能是管理、协调和监控在集群中运行的 Storm 应用程序，也就是 Topology。Supervisor 则负责接收 Nimbus 发送的 Topology 以及生成 Worker 来运行相应的 Task。ZooKeeper 的主要职责是在分布式环境下提供集中式信息维护管理服务，它可以运行在不同的集群中。在 Storm 集群中，ZooKeeper 主要用来提供集群状态信息的维护管理，数据的传输并不是 ZooKeeper 负责。

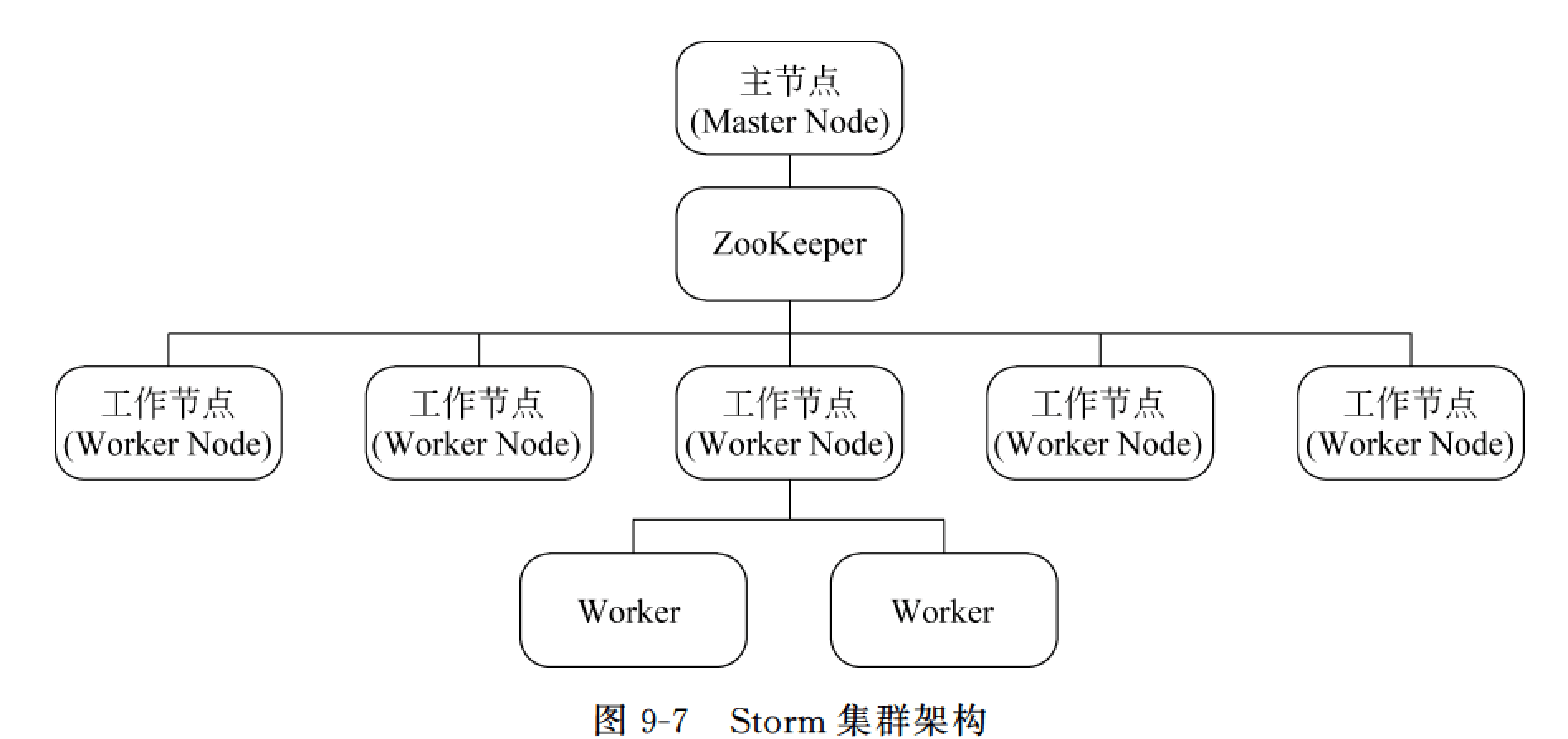


图 9-7　Storm 集群架构

2. Storm 安装部署

Storm 运行在 Java 虚拟机上，主要接口都是用 Java 语言编写的，但在开发 Storm 应用程序时（Spout 和 Bolt），用户也可以利用其他语言编写。另外，Storm 的后台程序和管

理命令都是用 Python 启动的，因此在安装 Storm 之前，必须先安装 Java(版本 Java 7)和 Python(版本 2.6.6)语言支持。

Storm 可以运行在 Linux 和 Windows 环境下，这里简要介绍 Storm 在 Linux(以 Ubuntu 为例)下的单节点伪集群安装部署。多节点安装部署及高级配置可以参考 Storm 官方文档，见参考文献[4]。Storm 的安装流程如下：

(1) 安装 ZooKeeper。

在终端(Terminal)中执行如下命令：

```
sudo apt-get -yes install zookeeper=3.3.5* zookeeperd=3.3.5*
```

此命令在目标系统中安装二进制版本的 ZooKeeper，并会生成一个脚本来启动和关闭 ZooKeeper。

(2) 安装 Storm。

首先下载 Storm 发行版(http://github.com/apache/storm/releases)并安装在/usr/share 目录下。在这里可以设定一个与版本无关的软连接(路径映射)，将 Storm 的可执行程序连接到/usr/bin/storm 目录下。

(3) 配置 Storm。

Storm 的配置文件是 conf/storm.yaml。在安装一个新的伪分布式 Storm 集群时，下列配置是需要重新定义的：

```
storm.zookeeper.servers:
    - "localhost"
nimbus.host: "localhost"
supervisor.slots.ports:
    - 6700
    - 6701
    - 6702
    - 6703
storm.local.dir: "/home/storm"
```

(4) 启动守护进程。

在终端执行如下命令启动 Nimbus 守护进程：

```
bin/storm nimbus
```

在终端执行如下命令启动 Supervisor 守护进程：

```
bin/storm supervisor
```

在终端执行如下命令启动 Storm UI：

```
bin/storm ui
```

至此，一个最基本的单节点伪分布式 Storm 集群安装完毕并运行起来了，用户可以从 http://localhost:8080 查看 Storm 是否已经启动运行。Storm 的运行日志可以在安装目录的 logs 文件夹下找到。

9.4.3 Storm 编程

本节用两个例子来学习如何进行 Storm 编程。Storm 编程主要分为两个步骤。第一个步骤为逻辑设计，即设计整个应用程序需要多少个 Spout，多少个 Bolt，Spout 与 Bolt 分别负责什么功能，Spout 与 Bolt 之间的拓扑链接关系如何。第二个步骤是编写代码实现这些功能。先来看一个单词计数程序。

1. 在线单词计数

在线单词计数的功能是从一个不停产生文本的数据源获取数据，并对其中包含的单词进行计数。与 MapReduce 不同，Storm 的数据源并非 HDFS 文件，而是实时数据源。一个简单的单词计数 Storm 程序的 Topology 如图 9-8 所示。

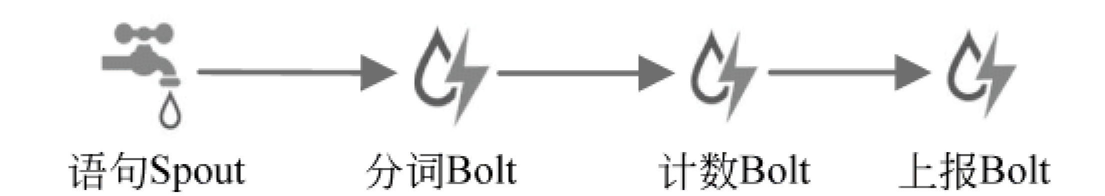


图 9-8 单词计数程序 Topology

各个组件的功能如下：

- 语句 Spout。获取数据源的文本，并向后端发射一个字符串，在本例中，使用一个静态字符串数组循环发射来模拟实时数据流。
- 分词 Bolt。获取语句 Spout 发射的字符串，并将字符串分割成单词的序列，向后发射。
- 计数 Bolt。获取分词 Bolt 发射的单词，并累计每个单词出现的次数，同时向后发射当前时刻该单词的计数。
- 上报 Bolt。获取计数 Bolt 发射的词语及计数，并在终端输出。

代码实现如下。

(1) SentenceSpout.java，实现语句 Spout。从一个数组循环读取字符串模拟实时数据流。字符串发射间隔为 10ms。发射的 Tuple 格式为{"sentence"："字符串内容"}。

```
public class SentenceSpout extends BaseRichSpout {
    private SpoutOutputCollector collector;
    private String[] sentences={
            "this is my first storm program",
            "it counts the words in sentences",
            "it is fast",
            "I enjoy programing with storm"
    };
    private int index=0;

    public void nextTuple() {
        this.collector.emit(new Values(sentences[index]));
        index++;
```

```
            index=index% sentences.length;
            try {
                Thread.sleep(1);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        public void open(Map config, TopologyContext context, SpoutOutputCollector
              collector) {
            this.collector=collector;
        }

        public void declareOutputFields(OutputFieldsDeclarer declarer) {
            declarer.declare(new Fields("sentence"));
        }
    }
```

（2）SplitBolt.java，实现分词Bolt，主要功能为接收句子Spout发来的字符串并以空格切分为单词，向后发射，发射格式为{"word"："单词内容"}。

```
    public class SplitBolt extends BaseRichBolt{
        private OutputCollector collector;

        public void execute(Tuple tuple) {
            String sent=tuple.getStringByField("sentence");
            String[] words=sent.split(" ");
            for(String w : words)
                collector.emit(new Values(w));
        }

        public void prepare(Map config, TopologyContext context, OutputCollector collector) {
            this.collector=collector;
        }

        public void declareOutputFields(OutputFieldsDeclarer declarer) {
            declarer.declare(new Fields("word"));
        }
    }
```

（3）CountBolt.java，实现计数Bolt，主要功能为接收分词Bolt发来的单词并利用一个HashMap计数，然后将计数结果向后发射，发射格式为{"word"："单词内容"；"count"：单词计数}。

```
    public class CountBolt extends BaseRichBolt{
```

```
    private OutputCollector collector;
    private HashMap< String, Long> counts;

    public void execute(Tuple tuple) {
        String word=tuple.getStringByField("word");
        Long c=1L;
        if(counts.containsKey(word))
            c +=counts.get(word);
        counts.put(word, c);
        this.collector.emit(new Values(word,c));
    }

    public void prepare(Map config, TopologyContext context, OutputCollector
    collector) {
        this.collector=collector;
        counts=new HashMap< String, Long> ();
    }

    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("word","count"));
    }
}
```

（4）PrintBolt.java，实现打印 Bolt，主要功能为打印最终结果到终端，这个 Bolt 并不向后发射任何 Tuple，因此其 declareOutputFields 方法无须具体实现。

```
public class PrintBolt extends BaseRichBolt{
    private HashMap< String, Long> counts;

    public void execute(Tuple tuple) {
        String word=tuple.getStringByField("word");
        Long c=tuple.getLongByField("count");;
        counts.put(word, c);
    }

    public void prepare(Map config, TopologyContext context, OutputCollector
    collector) {
        counts=new HashMap< String, Long> ();
    }

    public void cleanup(){
        System.out.println("Word Counts:");
        Iterator< Entry< String, Long>> it=this.counts.entrySet().iterator();
        while(it.hasNext()){
            Entry en=it.next();
```

```
                System.out.println(en.getKey() + "\t" + en.getValue());
            }
            System.out.println("-------------END----------");
        }
        public void declareOutputFields(OutputFieldsDeclarer declarer) {      }
    }
```

这里展示了一个 Storm 单词计数程序的主要组件(Spout，Bolt)的实现。要让这个程序在 Storm 运行，还需要设计 Topology，将这些组件连接起来，然后提交到 Storm。本例中 Topology 的主要代码实现如下：

```
public class WordCountTopology{
    private static final String SENTENCE_SPOUT_ID="sent-spout";
    private static final String SPLIT_BOLT_ID="split-bolt";
    private static final String COUNT_BOLT_ID="count-bolt";
    private static final String PRINT_BOLT_ID="print-bolt";
    private static final String TOPOLOGY_NAME="wordcount-topology";

    public static void main(String[] args) throws InterruptedException{
        SentenceSpout spout=new SentenceSpout();
        SplitBolt splitBolt=new SplitBolt();
        CountBolt countBolt=new CountBolt();
        PrintBolt printBolt=new PrintBolt();

        TopologyBuilder builder=new TopologyBuilder();
        builder.setSpout(SENTENCE_SPOUT_ID, spout,2);
        builder.setBolt(SPLIT_BOLT_ID, splitBolt,2).setNumTasks(4).
                shuffleGrouping(SENTENCE_SPOUT_ID);
        builder.setBolt(COUNT_BOLT_ID, countBolt,2).setNumTasks(4).
                fieldsGrouping(SPLIT_BOLT_ID, new Fields("word"));
        builder.setBolt(PRINT_BOLT_ID, printBolt,2).globalGrouping(COUNT_
                BOLT_ID);

        Config conf=new Config();
        LocalCluster cluster=new LocalCluster();
        cluster.submitTopology(TOPOLOGY_NAME, conf, builder.createTopology());

        Thread.sleep(10000);

        cluster.killTopology(TOPOLOGY_NAME);
        cluster.shutdown();

    }
}
```

在 Topology 的实现中，组件之间的关联关系以代码的方式体现出来了。例如：

```
builder.setBolt(COUNT_BOLT_ID, countBolt,2).setNumTasks(4).fieldsGrouping(SPLIT_BOLT_
ID, new Fields("word"));
```

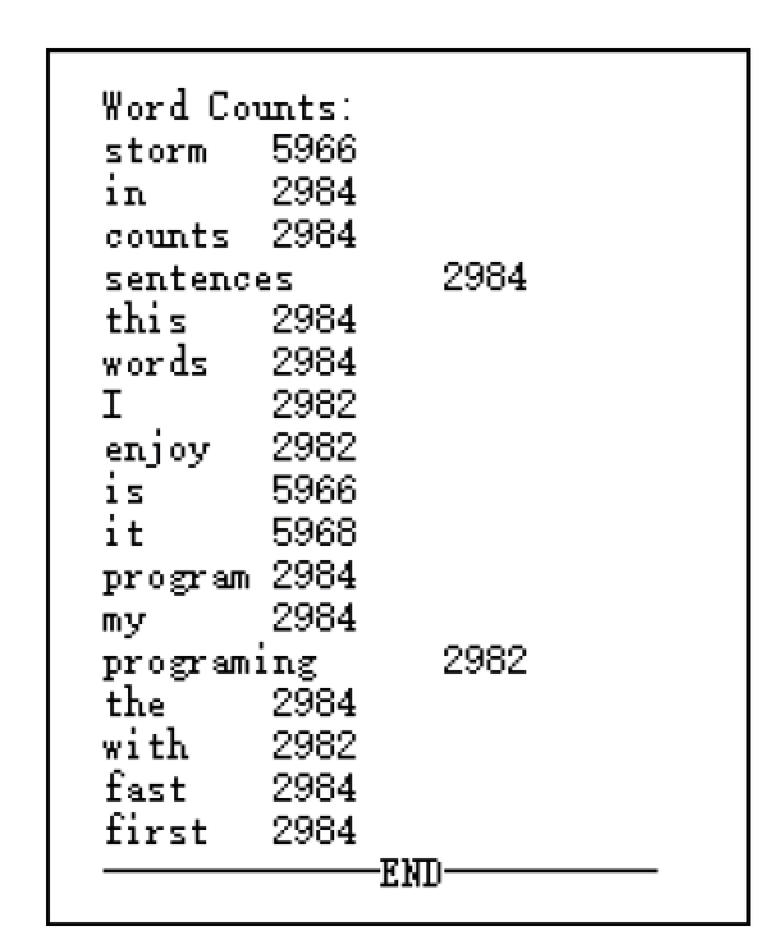
表示 CountBolt 从 SplitBolt 以 fieldsGrouping 的方式订阅了 Tuple 流，所有从 SplitBolt 发射出的 Tuple 都会被送到 CountBolt，fieldsGrouping 表示这些 Tuple 是根据 word 的值进行分组的，相同 word 值的 Tuple 会被分配到同一个 Task。参数“2”和 setNumTasks(4)表示 CountBolt 由 2 个 Excutor 生成 4 个 Task 执行。因此这个 Bolt 会产生 4 个线程并行进行计数。

最终程序的输出如图 9-9 所示。

在本例中，10s 内进行了约 2982 轮，也就是约 11 928 个句子的计数。考虑到每个句子发射出去后利用代码强制休眠了 1ms，这个实时性能还是比较高的。

2. 在线 SQL 注入检测

上面介绍了一个实时单词计数程序，下面模拟一个在线 SQL 注入检测的实例。同样采用一个循环 URL 字符串数组来模拟实时的 URL 请求。假定已经有了一个简单的 SQL 注入检测模型，如图 9-10 所示。



```
Word Counts:
storm    5966
in       2984
counts   2984
sentences        2984
this     2984
words    2984
I        2982
enjoy    2982
is       5966
it       5968
program  2984
my       2984
programing       2982
the      2984
with     2982
fast     2984
first    2984
------------END------------
```

图 9-9　程序输出结果

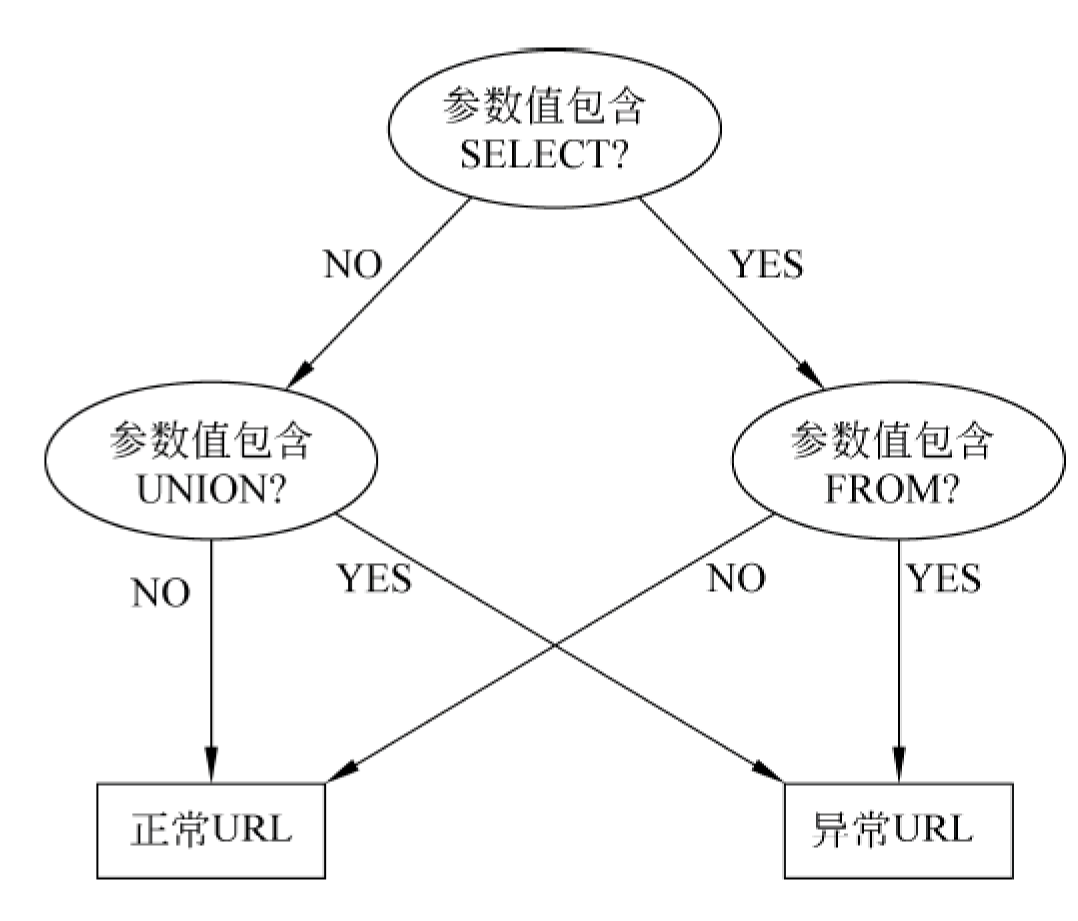


图 9-10　SQL 注入检测模型(样例)

图 9-10 展示了一个简单的 SQL 注入检测模型样例。如果 URL 参数值既包含 SELECT 也包含 FROM，则该 URL 异常，如果 URL 参数值不包含 SELECT 但包含 UNION，则该 URL 异常，否则该 URL 正常(请注意，这只是一个用于示例的检测模型，并不能真正用于检测 SQL 注入，实际应用中的 SQL 注入检测模型比这个要复杂得多)。

那么，如何将这个检测模型转换为一个 Storm 程序呢？

图 9-11 的设计思路是这样的：首先利用 URL Spout 获取实时 URL 并发射，然后参数提取 Bolt 对接收到的 URL 进行参数提取，对敏感特征 SELECT、FROM、UNION 分别构建域并存放在 Tuple 中，向后发射。SELECT_YES Bolt 是一个过滤功能的 Bolt，只

有参数值包含 SELECT 的才会被发射到后方，SELECT_NO Bolt 同理。因此虽然 SELECT_YES Bolt 和 SELECT_NO Bolt 获取到的 Tuple 流是一样的，但发射出去的 Tuple 流是不同的。FROM_YES Bolt、FROM_NO Bolt、UNION_YES Bolt、UNION_NO Bolt 同理。最后打印 Bolt 获取所有输出 Tuple 流并打印。

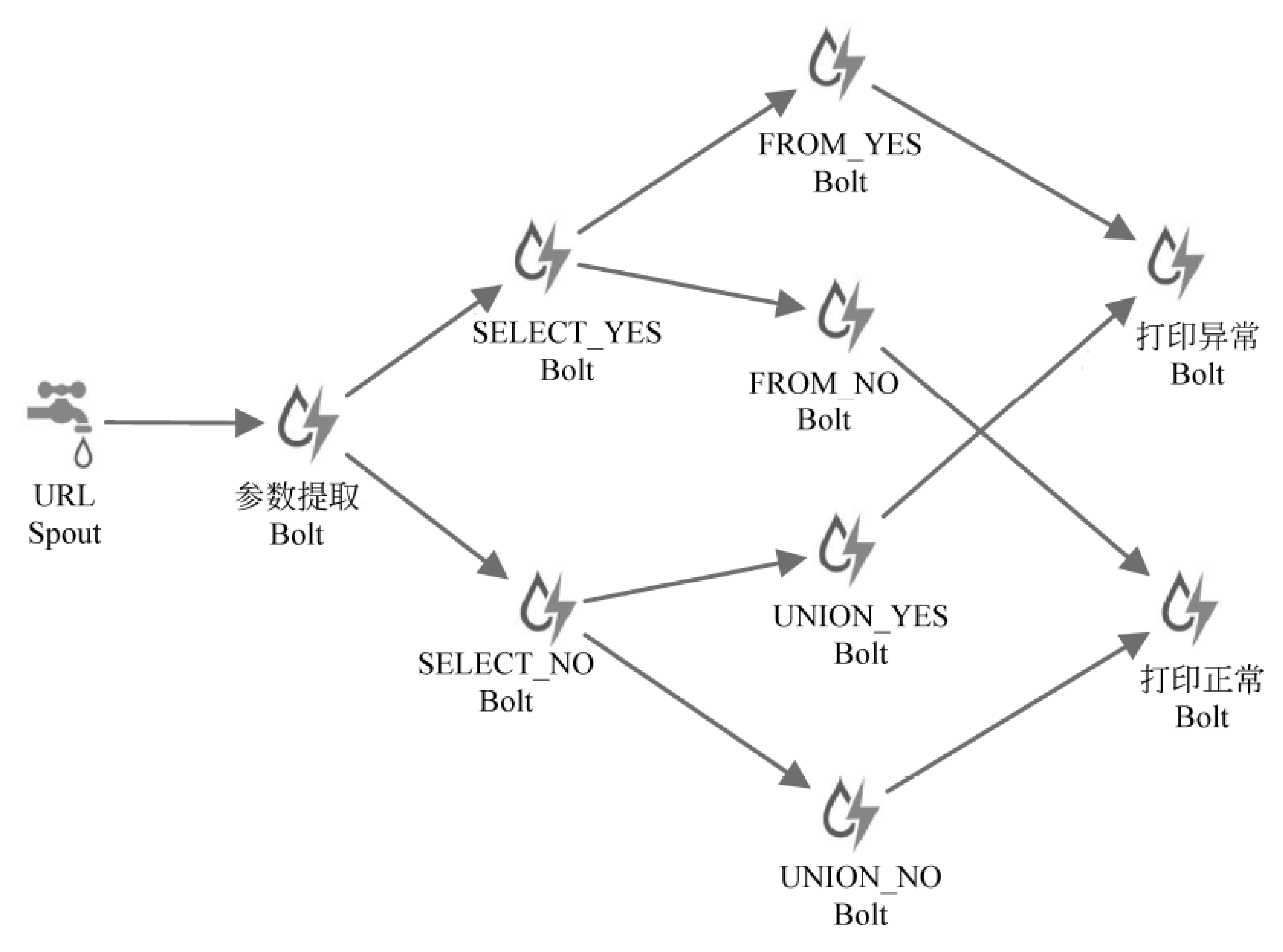


图 9-11　SQL 注入检测程序 Topology 设计

由于 URL Spout 与前例中的句子 Spout 基本类似，这里略去代码。

参数 Bolt 代码如下：

```
public class ParaBolt extends BaseRichBolt{
    private OutputCollector collector;

    public void execute(Tuple tuple) {
        String url=tuple.getStringByField("url");
        boolean containsSELECT=contains(url,"SELECT");
        boolean containsFROM=contains(url,"FROM");
        boolean containsUNION=contains(url,"UNION");
        collector.emit(new Values(url, containsSELECT, containsFROM,
                containsUNION));
    }

    public void prepare(Map config, TopologyContext context, OutputCollector
          collector) {
        this.collector=collector;
    }
```

```
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("URL","SELECT","FROM","UNION"));
    }

    public boolean contains(String url,String str){
        return url.contains(str);
    }
}
```

SELECT_YES Bolt 代码如下：

```
public class SelectYesBolt extends BaseRichBolt{
    private OutputCollector collector;

    public void execute(Tuple tuple) {
        String url=tuple.getStringByField("url");
        boolean containsSELECT=tuple.getBooleanByField("SELECT");
        boolean containsFROM=tuple.getBooleanByField("FROM");
        boolean containsUNION=tuple.getBooleanByField("UNION");
        if(containsSELECT)
            collector.emit(new Values(url, containsSELECT, containsFROM,
                containsUNION));
    }

    public void prepare(Map config, TopologyContext context, OutputCollector
          collector) {
        this.collector=collector;
    }

    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("URL","SELECT","FROM","UNION"));
    }
}
```

SELECT_NO Bolt、FROM_YES Bolt、FROM_NO Bolt、UNION_YES Bolt、UNION_NO Bolt 的实现与 SELECT_YES Bolt 类似，在此略去。

打印正常 URL 的 Bolt 的实现如下：

```
public class PrintSQLBolt extends BaseRichBolt{
    private ArrayList<String> urls;

    public void execute(Tuple tuple) {
        String url=tuple.getStringByField("url");
        urls.add(url);
    }
```

```
    public void prepare(Map config, TopologyContext context, OutputCollector
            collector) {
        urls=new ArrayList<String>();
    }

    public void cleanup(){
        for(String url : urls){
            System.out.println("SQL INJECTION URL:\t"+url);
        }
    }
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
    }

}
```

打印正常 URL 的 Bolt 实现类似，在此略去。

最关键的部分是 Topology 的实现，这一部分主要在于构建 Spout 与 Bolt 之间的链接关系，关键代码如下：

```
public class SQLTopology{
    …
    public static void main(String[] args) throws InterruptedException{
        …
        TopologyBuilder builder=new TopologyBuilder();
        builder.setSpout(URL_SPOUT_ID, urlSpout);
        builder.setBolt(PARA_BOLT_ID, paraBolt) .shuffleGrouping(URL_SPOUT_ID);
        …
        builder.setBolt(PRINT_SQL_BOLT_ID, printSqlBolt). globalGrouping
                (UNION_YES_BOLT_ID).globalGrouping (FROM_YES_BOLT_ID);
        builder.setBolt(PRINT_NOR_BOLT_ID, printNorBolt). globalGrouping
                (UNION_NO_BOLT_ID).globalGrouping (FROM_NO_BOLT_ID);
        …
    }
}
```

这里省略了字符串常量、Spout、Bolt 以及若干中间 Bolt 连接关系的定义。程序最终输出如图 9-12 所示，这些是我们构造的实验用的 URL，输出与期望的结果相符。

```
SQL INJECTION URL:      http://www.excample.com/detail.jsp?id=1'+(SELECT 1 FROM (SELECT SLEEP(5))A)+'
SQL INJECTION URL:      http://www.excample.com/detail.jsp?id=1 and 1=2 UNION 123
SQL INJECTION URL:      http://www.excample.com/detail.jsp?id=1'+(SELECT 1 FROM (SELECT SLEEP(5))A)+'
SQL INJECTION URL:      http://www.excample.com/detail.jsp?id=1 and 1=2 UNION 123
SQL INJECTION URL:      http://www.excample.com/detail.jsp?id=1'+(SELECT 1 FROM (SELECT SLEEP(5))A)+'
NORMAL URL:     http://www.excample.com/detail.jsp?id=123
NORMAL URL:     http://www.excample.com/search.jsp?keyword=select
NORMAL URL:     http://www.excample.com/detail.jsp?id=123
NORMAL URL:     http://www.excample.com/search.jsp?keyword=select
```

图 9-12　程序输出结果

9.5 基于 Spark Streaming 的分布式实时计算

在 9.4 节中介绍了如何利用 Storm 进行实时流数据分析计算。本节介绍另外一个实时流数据分析计算引擎。

9.5.1 Spark 内存计算框架

内存计算是以大数据为中心,依托计算机硬件的发展,依靠新型的软件体系结构,通过将数据装入内存中处理,而尽量避免 I/O 操作的一种新型的以数据为中心的并行计算模式。在应用层面,内存计算主要用于数据密集型计算的处理,尤其是数据量极大且需要实时分析处理的计算。这类应用以数据为中心,需要极高的数据传输及处理速率。因此,在内存计算模式中,数据的存储与传输取代了计算任务成为新的核心。

分布式内存处理系统主要处理机器学习、图算法、科学计算等问题,此类问题在并行计算的同时,往往涉及结构或逻辑上的依赖关系,而并行处理的各个步骤到达稳定点的时间不同,因此需要在并行的计算步之间进行同步控制,以保证结果的正确性。常见的同步方式有同步计算、异步计算和混合方式。目前,大多数内存计算系统采用 BSP 同步,部分系统采用异步机制。Spark 和 Pregel 都采用 BSP 同步机制,而基于内存计算的分布式内存共享图处理系统 PowerGraph 则采用 BSP 同步和异步两种方式。Trinity 同样采用 BSP 同步和异步两种方式。典型内存计算框架的同步方式如表 9-10 所示。

表 9-10 典型内存计算框架的同步方式

内存计算框架	同步方式	内存计算框架	同步方式
Spark	同步计算机制	PowerGraph	混合计算机制
Pregel	同步计算机制	Trinity	混合计算机制

MapReduce 计算框架模型简单,实现了 DAG(有向无环图)的 data flow 式的计算,不能有效处理有环的计算,也就是输入同时作为输出的循环计算。Spark 主要解决的问题就是在当前的分布式计算框架中不能有效处理迭代计算和交互式计算两类问题。Spark 更适合于迭代运算比较多的机器学习和数据挖掘运算,这也是其目前受到广泛关注的原因。Spark 将内存数据抽象成 RDD(Resilient Distributed Dataset,弹性分布式数据集),然后在内存不足时,利用“最近最少使用”(LRU)内存替换策略协调内存资源。同时,为了更灵活地分配内存资源,Spark 可以通过所谓的 RDD“持续存储优先权”给用户一定的管理权限。

那么 Spark 是如何实现的呢?其主要的思想就是构建一个 RDD 把所有计算的数据保存在分布式的内存中。在迭代计算中,通常情况下,都是对同一数据集做反复的迭代计算,数据保存在内存中,将大大提高性能。RDD 模型的产生动机主要来源于两种主流的应用场景:

- 迭代式算法。如迭代式机器学习、图算法,包括 PageRank、k-means 聚类和逻辑回归。

• 交互式数据挖掘工具。用户在同一数据子集上运行多个 Adhoc 查询。

不难看出,这两种场景的共同之处是:在多个计算或计算的多个阶段间重用中间结果。在之前学习的 MapReduce 计算框架中,要想在计算之间重用数据,唯一的办法就是把数据保存到外部存储系统中,即中间结果写入本地磁盘。这就导致了巨大的数据复制、磁盘 I/O、序列化的开销,甚至会占据整个应用执行时间的一大部分。

下面以 MapReduce 中的经典例子 WordCount 为例解释 Spark 执行模型,整个 WordCount Job 是计算 README.md 这个文档中各个单词出现的频次,并且将最终结果保存到 wcresult 目录中去。下面以 WordCount 例子来说明 Spark 执行模型。

```
val wordCountResult=sc.textFile("README.md",4).flatMap(line=>line.split(" "))
        .map(word=>(word,1)).reduceByKey(_+_,2)
    wordCountResult.saveAsTextFile("wcresult")
```

从 RDD DAG 的角度来看,该 RDD DAG 主要是包括 MappedRDD→FlatMappedRDD→MappedRDD→ShuffledRDD 这 4 个 RDD 的转换操作(transform),如图 9-13 所示。根据 Spark 实现,RDD 的转换操作是不会提交给 Spark 集群来执行的,因此,上面的操作必须由 Spark 的执行操作(action)来触发,因此,在最后调用 saveAsTextFile 这个行为来将整个 WordCount Job 提交到 Spark 集群中来执行。

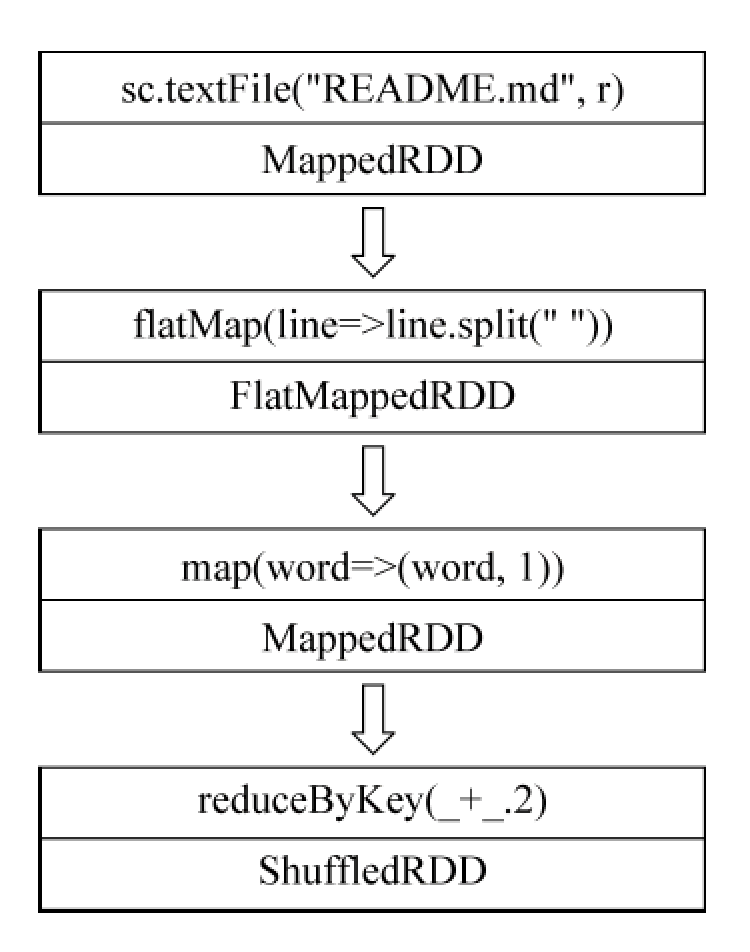


图 9-13 WordCount Job 的 RDD DAG 图

9.5.2 Spark Streaming 简介

Spark Streaming 是 Spark 核心 API 的一个扩展,用于支持可扩展、高吞吐率、高容错性的实时流数据分析处理。数据源可以是多样的,如 Kafka、Flume、Kinesis、TCP sockets 等。这些数据可以利用 Spark 自带的各类算子如 map、reduce、join、window 等进行处理。最后,经过处理的数据可以输出到文件系统、数据库、实时仪表盘等。图 9-14 展示了 Spark Streaming 的数据处理流程。



图 9-14 Spark Streaming 数据处理流程

图片来源 http://spark.apache.org/docs/latest/streaming-programming-guide.html

在 Spark 中,数据并非到达即处理,而是将数据划分为一个一个的块(bacth),然后以

块为单位处理。因此 Spark 只能算是准实时的流数据计算，但这样可以提高效率。块的划分越细，实时性越强，一般来说效率越低；反之则实时性降低，效率提高。图 9-15 展示了 Spark Streaming 分块对流数据计算的过程：

实时流数据
x秒的块
Spark Streaming
Spark Engine
处理后的结果

图 9-15　Spark Streaming 对流数据进行分块计算

1. Spark Streaming 基本概念

Spark Streaming 并非一个独立的项目，它的计算引擎依赖于 Spark，因此 Spark Streaming 实时计算只有在部署了 Spark 的计算机上才能运行。限于篇幅，这里不再介绍 Spark 的安装部署，感兴趣的读者可以到 Spark 官方网站 http://spark.apache.org/docs/latest/查阅相应的文档。这里仅对相关的若干基本概念进行介绍。

1）弹性分布式数据集（RDD）

RDD 是 Spark 的核心数据结构，它是一个高容错性的可并行处理元素的集合。可以通过两种方式建立 RDD：一是将程序中的数据集合直接并行化，二是从外部存储系统中引用得到。在建立 RDD 之后，就可以以并行的方式来操作 RDD 中存储的数据元素。

2）离散化数据流（DStream）

DStream 是 Spark Streaming 的核心数据结构，其本质是一个 RDD 序列。这与 Storm 不同，在 Storm 中，数据流的基本单元是 Tuple，也就是数据的最小粒度，但在 Spark Streaming 中，DStream 的基本单元是 RDD，也就是一个数据块（batch）。直观上来看，就像是将连续的数据流切分成若干数据块的序列，因此被称为离散化数据流。图 9-16 展示了 DStream 的构成。

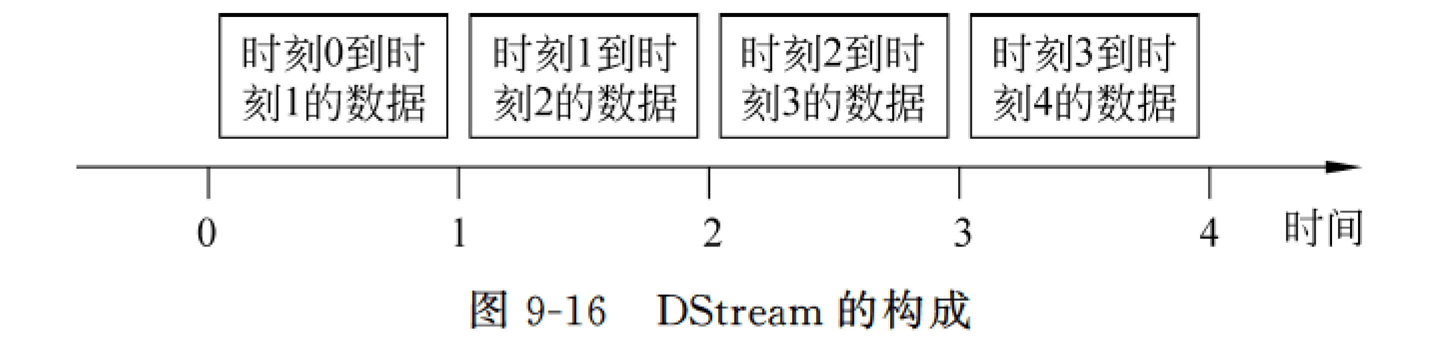


图 9-16　DStream 的构成

2. Spark Streaming 基本操作

Spark 程序的运行过程实际上就是对 RDD 的转换过程。同样，Spark Steaming 程序的运行过程实际上就是对 DStream 的转换过程。用户无须关心如何将这些转换并行化，只需要考虑如何利用 Spark Streaming 的操作实现程序的功能。常用操作如表 9-11 所示。

DStream 还有更多其他的操作，限于篇幅这里不一一阐述，读者可以到 Spark Streaming 的官网阅读参考文献[11]。

表 9-11 DStream 基本操作

转换算子	功　能
map(func)	利用 func 将源 DStream 转换为一个新的 DStream 并返回
flatMap(func)	与 map 相似，但每个输入项都可以映射为 0 个或多个输出项
filter(func)	将源 DStream 中符合 func 所述条件的项选择出来并组成一个新的 DStream 并返回
repartition(numPartitions)	通过创建更多或更少划分更改当前 DStream 的并行化等级
union(otherStream)	将当前 Dstream 与 otherDStream 进行 union 操作并生成一个新的 DStream 返回
count()	返回一个单元素 RDD 组成的 DStream。该 DStream 中每一个 RDD 的元素的内容为源 DStream 相应 RDD 所包含的元素的个数
reduce(func)	返回一个单元素 RDD 组成的 DStream。该 DStream 中每一个 RDD 的元素的内容为通过 func 将源 DStream 中的相应的 RDD 的元素进行聚合得到。func 输入参数为 2 个，输出为 1 个，必须可并行计算
countByValue()	对源 DStream 中每个 RDD 的值进行计数，并生成一个新的 DStream 返回
reduceByKey(func, [numTasks])	对形如(K, V)对的 DStream 进行计算，并返回一个新的(K, V)对组成的 DStream，新 Dstream 中的 V 为源 DStream 中相应 RDD 根据键值聚合而成的
join(otherStream, [numTasks])	对形如(K, V) 与 (K, W) 对的两个 DStream 进行 join 操作，返回一个新的形如(K, (V, W)) 对的 DStream
cogroup(otherStream, [numTasks])	对形如(K, V) 与 (K, W) 对的两个 DStream 进行 group 操作，返回一个新的形如(K, Seq[V], Seq[W]) 三元组的 DStream
transform(func)	利用 RDD 操作将一个 DStream 的每一个 RDD 转换为一个新的 RDD 并组成新的 DStream 返回
updateStateByKey(func)	返回一个新的状态 DStream，新 DStream 中每个键的状态由源 DStream 中每个键的状态通过 func 得到

9.5.3 Spark Streaming 编程

本节学习如何使用 Spark Streaming 实现流数据分析。与 Storm 相同，这里也采用在线单词计数和在线 SQL 注入检测来作为例子进行阐述。

由于 Spark Streaming 没有类似于 Storm 中 Spout 这样的数据发生器组件，因此这里采用 Socket 流数据作为输入。Netcat 是一个 UNIX/Linux 平台下的 TCP/IP 工具，可以用于产生 Socket 流，用户可以到 http://netcat.sourceforge.net/下载安装。这个工具也有 Windows 版，用户可以到 https://www.joncraton.org/blog/46/netcat-for-windows/下载安装。安装完毕后，运行如下命令，就可打开本机上的 6666 端口并监听，等待连接。

```
nc -l -p 6666
```

建立连接后，在 Netcat 运行终端输入的字符串都会被发送到连接方。在在线单词计数程序中，可以在 Netcat 终端输入不同的句子作为实时数据；在在线 SQL 注入检测程序

中，每一行输入一个异常或正常的 URL。

1. 在线单词计数

由于利用了 Spark 核心组件的功能，Spark Streaming 在线单词计数的程序相对较为简洁，主要分为 3 个部分。

- 配置 Spark 集群并启动。
- 初始化 Socket 组件，并将 Socket 流转换为 DStream。
- 利用 DStream 转换操作对句子中的单词进行计数。

配置 Spark 集群的代码如下：

```
SparkConf sparkConf=new SparkConf().setAppName("WordCount");
sparkConf.setMaster("local[4]");
```

第一行代码配置应用的名称为 WordCount，第二行代码表示当前应用以本地模式 4 线程方式运行，

初始化 Socket 组件 JavaStreamingContext 代码如下：

```
String hostname="localhost";
String port="6666";
JavaStreamingContext ssc=new JavaStreamingContext(sparkConf,
                    Durations.seconds(1));
JavaReceiverInputDStream<String> lines=ssc.socketTextStream(
    hostname, Integer.parseInt(port), StorageLevels.MEMORY_AND_DISK_SER);
```

这段代码初始化了一个 JavaStreamingContext 从 localhost：6666 以 socket 方式获取文本流。切分间隔为 1s，也就是说 1s 之内的所有文本会被放到一个 RDD 内。变量 lines 就是一个由这些 RDD 序列组成的 DStream。要增加块的大小，可以增加 Durations. seconds(n)中 n 的大小。这样可以提高吞吐率，但会降低实时性。Durations 对象还有 miniseconds 方法，可以以毫秒为单位切分流数据，提高实时性，但会降低吞吐率。

单词计数代码如下：

```
JavaDStream<String> words=lines.flatMap(new FlatMapFunction<String, String>(){
  @Override
  public Iterator<String> call(String x) {
    return Arrays.asList(SPACE.split(x)).iterator();
  }
});
JavaPairDStream<String, Integer> wordCounts=words.mapToPair(
  new PairFunction<String, String, Integer>() {
    @Override
    public Tuple2<String, Integer> call(String s) {
      return new Tuple2<>(s, 1);
    }
  }).reduceByKey(new Function2<Integer, Integer, Integer>() {
```

```
        @Override
        public Integer call(Integer i1, Integer i2) {
          return i1 + i2;
        }
      });
```

在这一段代码中，lines 变量首先被 flatMap 操作转换为 words，一个以 String 为基本类型的 DStream。转换方式为以空格切分句子。然后单词 DStream 通过 mapToPair 操作转换为<word, 1>这样的二元组，最后以 word 为键进行 reduceByKey 转换，得到每个单词的计数。

当在 Netcat 终端输入"this is my first spark streaming program, I love coding with spark streaming"后，程序的结果如图 9-17 所示。

```
-------------------------------------------
Time: 1475765733000 ms
-------------------------------------------
(first,1)
(my,1)
(program,,1)
(spark,2)
(I,1)
(this,1)
(is,1)
(coding,1)
(love,1)
(with,1)
...
```

图 9-17 Spark Streaming 单词计数结果

2. 在线 SQL 注入检测

在这里利用 Spark Streaming 重新编写 9.4.3 节中的在线 SQL 注入检测程序。同样采用图 9-10 所表示的样例检测模型。与上面的例子相同，这里也采用 Netcat 作为数据源。测试样例如下：

```
http://www.excample.com/detail.jsp?id=1'+(SELECT 1 FROM (SELECT SLEEP(5))A)+'
http://www.excample.com/detail.jsp?id=1 and 1=2 UNION 123
http://www.excample.com/detail.jsp?id=123
http://www.excample.com/search.jsp?keyword=select
```

前两个 URL 是有 SQL 注入嫌疑的 URL，后两个则是正常 URL。

检测程序的 Spark 配置、JavaStreamContext 初始化以及 lines DStream 的构建与上面的在线单词计数程序无异，主要区别在于获取数据之后的处理。检测代码如下：

```
JavaPairDStream<String, Boolean> results=lines.mapToPair(
    new PairFunction<String, String, Boolean>() {
      @Override
      public Tuple2<String, Boolean> call(String x) {
```

```
            String str=x.substring(x.indexOf("?"));
            String[] items=str.split("&");
            boolean containsSelect=false;
            boolean containsFrom=false;
            boolean containsUnion=false;
            for(String item : items){
                String paraValue=item.substring(item.indexOf("="));
                if(paraValue.contains("SELECT"))
                    containsSelect=true;
                if(paraValue.contains("FROM"))
                    containsFrom=true;
                if(paraValue.contains("UNION"))
                    containsUnion=true;
            }
            boolean isSQLInjection=false;
            if(containsSelect){
                if(containsFrom)
                    isSQLInjection=true;
            }
            else if (containsUnion)
                isSQLInjection=true;

            return new Tuple2<> (x, isSQLInjection);
        }
    });
```

这里定义了一个 results DStream 存储检测结果。这个 DStream 的基本单元是二元组,分别存放 URL 以及判定结果,若 URL 异常,则判定结果为 true,否则为 false。

程序主体部分定义了 3 个布尔变量,分别表示 URL 的参数值部分是否包含 SELECT、FROM 以及 UNION。然后用 if 判定语句实现图 9-10 所描述的模型,得到最终结果并存入二元组。程序的执行结果如图 9-18 所示。

```
-------------------------------------------
Time: 1475767453000 ms
-------------------------------------------
(http://www.excample.com/detail.jsp?id=1'+(SELECT 1 FROM (SELECT SLEEP(5))A)+',true)
(http://www.excample.com/detail.jsp?id=1 and 1=2 UNION 123,true)
(http://www.excample.com/detail.jsp?id=123,false)
(http://www.excample.com/search.jsp?keyword=select,false)
```

图 9-18 SQL 注入检测输出结果

9.6 小　　结

本章介绍了 Hadoop 的基础知识,介绍了 HDFS、MapReduce、YARN 的功能以及基本的配置与操作。

Hadoop 大数据分析处理平台的配置主要分本地模式、伪分布式模式与集群模式 3

种,分别对应着不同的配置方法。其中,本地模式常用于程序开发与调试,伪分布式模式常用于在本地模拟集群,集群模式主要用于生产作业。

HDFS是Hadoop平台下的分布式文件系统,主要负责集群中文件的存取管理,用户可以通过FS shell对HDFS中的文件进行操作。

MapReduce则是Hadoop平台的分布式计算引擎,通过构建Mapper与Reducer,编程人员可以忽略程序底层的并行计算与调度,将精力集中在具体的计算任务上,降低并行程序的编写难度,提高并行程序的编写效率。Hadoop Streaming则进一步地降低了编程人员的学习成本,可以使用任意程序语言或脚本编写MapReduce程序。

实时流数据计算是大数据分析处理的一个重要应用场景,本章介绍了两个实时流数据计算平台,分别是Storm以及Spark Streaming,应用程序对应的分布式计算结构称为Topology,包含若干个数据发生器Spout和数据运算单元Bolt。Spark Streaming则通过DStream的转换操作来对流数据进行计算。它们的不同之处在于,Storm对流数据的最小单元进行计算,Spark Streaming则是先对流数据进行切块处理,然后以块作为最小单元进行计算。

Hadoop发展至今,已经不仅仅是一个分布式数据存储与计算框架。以Hadoop为中心,衍生了诸多用于大数据分析与处理的项目,比如:

- HBase。一个用于大数据的分布式、可扩展的数据库系统,可以支持十亿级别记录与百万级别属性的结构化大数据表(Bigtable)的存储与管理,支持严格的读写一致性管理,可以使用Java API进行访问。
- Hive。一个支持随机查询的数据仓库工具,可以将结构化的数据文件映射为数据库表并提供基本的SQL查询功能。值得一提的是Hive可以将用户提交的SQL语句转化为MapReduce程序执行,在降低用户学习成本的同时大大提高了程序的执行效率。
- Mahout。一个基于MapReduce的支持分布式计算的机器学习工具库,包含若干常用的数据挖掘与机器学习算法的MapReduce实现(目前主要倾向于Spark实现)。

Hadoop生态环境还包含若干其他工具,如Pig、Zookeeper等,每一个组件或工具都有其独特的功能和特点,读者可以根据需要有选择地学习。

9.7 参考文献

[1] Hadoop和Java版本之间的兼容性说明. http://wiki. apache. org/hadoop/HadoopJavaVersions.

[2] Hadoop集群配置. http://hadoop. apache. org/docs/r2. 7. 0/hadoop-project-dist/hadoop-common/ClusterSetup. html.

[3] Hadoop安全模式配置. http://hadoop. apache. org/docs/r2. 7. 0/hadoop-project-dist/hadoop-common/SecureMode. html.

[4] Storm官方文档. http://storm. apache. org/releases/1. 0. 2/index. html.

[5] 罗乐,刘轶,钱德沛. 内存计算技术研究综述[J]. 软件学报,2016,27(8):2147-2167.

[6] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A System for Large-Scale Graph Processing. In: Proc. of the 2010 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2010: 135-

146. [doi: 10.1145/1807167.1807184]

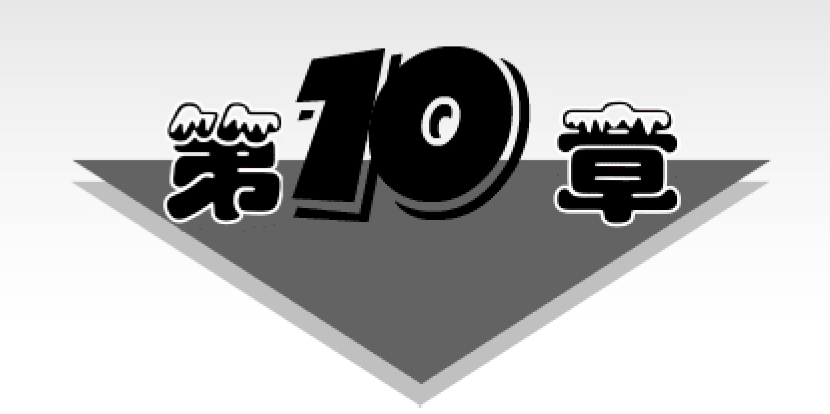
[7] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster Computing with Working Sets. In: Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing (HotCloud 2010). Berkeley: USENIX Association, 2010: 10-10.

[8] Valant L G. A Bridging Model for Parallel Computation. Communications of the ACM, 1990, 33 (8): 103-111. [doi: 10.1145/79173.79181]

[9] Gonzalez J E, Low Y, Gu H, et al. PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. In: Proc. of the 10th USENIX Conf. on Operating Systems Design and Implementation. Hollywood, 2012: 17-30.

[10] Shao B, Wang H, Li Y. Trinity: A Distributed Graph Engine on a Memory Cloud. In: Proc. of the 2013 Int'l Conf. on Management of Data. ACM Press, 2013: 505-516. [doi: 10.1145/2463676.2467799]

[11] Spark Streaming 官方文档. http://spark.apache.org/docs/latest/streaming-programming-guide.html.



大数据分析算法的并行化

在计算机技术发展的最初十几年里，从体系结构到系统软件以及应用软件都采用串行计算作为主要的设计和开发模式，但是很快人们就意识到，并行计算是突破串行计算效率瓶颈、提高计算性能的有力和必要的手段，相对于串行计算，并行计算可以划分为以流水线技术为代表的时间并行，以及以多处理器并发执行为代表的空间并行，两种并行方式都可以有效地提高计算资源的利用能力，从而提高程序的执行性能，因此并行计算的思想也开始渗透到计算机技术发展的各个方面。对于大多数程序员而言，早期编写分布式并行应用程序具有极高的编程门槛，不但需要程序员具有丰富的硬件、体系结构、操作系统等背景知识，还需要其编写大量复杂烦琐的、应用逻辑之外的分布式并行控制逻辑。如今在大数据时代，上述制约因素正在发生改变，由于云计算、大数据为分布式并行计算提供了各种新的高可定制的集群环境、编程模型及框架，为分布式并行计算在非科学计算领域的普及与应用带来了新的可能性和可行性。

本章介绍并行计算算法的基本知识和概念(10.1 节)，以 k-means 算法为典型案例分析其在 MapReduce 计算框架下的基本原理，并基于 Mahout 和 MLlib 进行并行化实现及应用讲解(10.2 节)，最后给出 3 个完整的 MapReduce 平台下数据分析的具体案例，使大家能够直接体会在大数据平台下进行数据分析的整个过程。

10.1 并行算法设计基础

本节给出并行算法相关的基本概念及应用术语，为读者学习并行计算及其编程实现建立基础。首先介绍了并行算法的概念，然后介绍并行计算的模型，讲解并行算法设计的策略和技术。

10.1.1 并行算法概念

并行计算或称平行计算是相对于串行计算来说的，是指同时使用多种计算资源解决计算问题的过程，是提高计算机系统计算速度和处理能力的一种有效手段。它的基本思想是用多个处理器来协同求解同一问题，即将被求解的问题分解成若干个部分，各部分均由一个独立的处理机来并行计算。并行计算系统既可以是专门设计的、含有多个处理器的超级计算机，也可以是以某种方式互连的若干台的独立计算机构成的集群。通过并行计算集群完成数据的处理，再将处理的结果返回给用户。并行算法就是用多台处理机联

合求解问题的方法和步骤，其执行过程是将给定的问题首先分解成若干个尽量相互独立的子问题，然后使用多台计算机同时求解它，从而最终求得原问题的解。

通常，可利用并行计算解决的问题一般表现出以下特征：

(1) 可将工作分离成离散部分，以有助于同时解决。

(2) 可随时并及时地执行多个程序指令。

(3) 多计算资源下解决问题的耗时应少于单个计算资源下的耗时。

定义 10.1　并行算法(parallel algorithm)是一些可同时执行的各个进程的集合，这些进程互相作用和协调动作，从而达到给定问题的求解。

所谓并行算法，可以朴素地解释为适合于在各种并行计算模型上求解问题和处理数据的算法。按照处理对象的类型，可以分为数值计算的并行算法和非数值计算的并行算法；按照进程的通信方式，又可以分为同步并行算法、异步并行算法以及分布并行算法。

定义 10.2　数值计算(numerical computation)是指基于代数关系的运算，包括矩阵运算、多项式求解、解线性方程组等计算问题，属于数值分析的范畴。

定义 10.3　非数值计算(non-numerical computation)是指基于比较关系的运算，包括排序、选择、搜索、匹配以及图论等方面的计算问题，属于符号处理的范畴。

定义 10.4　同步算法(synchronized algorithm)是指某些进程必须等待别的进程的并行算法。

定义 10.5　异步算法(asynchronized algorithm)是指各个进程的执行一般不必互相等待的并行算法，进程的通信通过动态读取(修改)共享存储器的全局变量实现。

定义 10.6　分布算法(distributed algorithm)是指由通信链路连接的多个场点或节点协同完成问题求解的并行算法。

10.1.2　并行计算模型

任何算法最终总要通过并行编程在具体的并行计算机上执行实现。本节简要讨论传统并行计算框架下的算法编程模型和大数据时代的并行计算模型。

1. 并行算法编程模型

1) 数据并行模型

数据并行模型既可以在SIMD(Single Instruction Multiple Data，单指令多数据流)计算机上实现，也可以在SPMD(Single Program Multiple Data，单程序多数据流)计算机上实现。数据并行模型的特点如下：

- 单线程。从程序员的观点看，一个数据并行程序中有一个进程执行，具有单一控制线；就控制流而论，一个数据并行程序就像一个顺序程序一样。
- 并行操作于集合数据结构上。数据并行程序的一个单步(语句)可指定同时作用在不同数据组元素或其他聚合数据结构上的多个操作。
- 松散同步。在数据并行程序的每条语句之后均有一个隐含的同步，这种语句级的同步是松散的(相对于SIMD计算机每条指令之后的紧同步而言)。
- 全局命名空间。数据并行程序中的所有变量均驻留在单一地址空间内，所有语句

可以访问任何变量，只需要满足通常的变量作用域规则即可。

- 隐式相互作用。因为数据并行程序的每条语句之末存在着一个隐含的路障，所以不需要一个显式同步，通信可由变量指派而隐含地完成。
- 隐式数据分配。程序员不必明确地指明如何分配数据，可将改进数据局部性和减少通信的数据分配方法揭示给编译器。

2）消息传递模型

在消息传递模型中，驻留在不同节点上的进程可以通过网络传递消息（指令、数据、同步信号或中断信号等）互相通信。

消息传递模型的特点如下：

- 多线程。消息传递程序由多个进程组成，每个进程都有其自己的控制线且可执行不同的代码；控制并行和数据并行均可支持。
- 异步并行性。消息传递程序的各个线程彼此异步地执行，使用诸如路障和阻塞通信的方法来同步各个进程。
- 分离地址空间。并行程序的进程驻留在不同地址空间内。一个进程中的数据变量在其他进程是不可见的，因此一个进程不能读/写另一进程中的变量，进程的相互作用通过执行特殊的消息传递操作实现。
- 显式相互作用。程序员必须解决包括数据映射、通信、同步和聚合等相互作用问题。
- 显式分配。负载和数据均由用户显式地分配给进程，为了降低设计和编码的复杂性，用户通常使用单一代码方法编写 SPMD 应用程序。

3）共享变量模型

在共享变量模型中，驻留在各处理器上的进程可以通过读/写公共存储器中的共享变量相互通信。它与数据并行模型的相似之处在于它有一个单一的全局名字空间，它与消息传递模型的相似之处在于它是多线程的和异步的。然而，数据是驻留在单一共享地址空间中的，因此不需要显式分配数据，而工作负载既可显式也可隐式分配。通信通过共享的读/写变量隐式完成，而同步必须是显式的，以保持进程执行的正确顺序。

以上 3 种典型并行算法编程模型的对比如表 10-1 所示。

表 10-1 典型并行算法编程模型对比

特征	数据并行	共享变量	消息传递
典型代表	HPF	OpenMP	MPI，PVM
可移植性	SMP，DSM，MPP	SMP，DSM	所有流行并行计算机
并行粒度	进程级细粒度	线程级细粒度	进程级大粒度
并行操作方式	松散同步	异步	异步
数据存储模式	共享存储	共享存储	分布式存储
数据分配方式	半隐式	隐式	显式
学习入门难度	偏易	容易	较难
可扩展性	一般	较差	好

MPI(Message Passing Interface,消息传递接口)是由全世界工业、科研和政府部门联合建立的一个消息传递编程标准,其目的是为基于消息传递的并行程序设计提供一个高效、可扩展、统一的编程环境。它是目前最为通用的并行编程方式,也是分布式并行系统的主要编程环境。MPI 标准中定义了一组函数接口用于进程间的消息传递。这些函数的具体实现由各计算机厂商或科研部门完成。除各厂商提供的 MPI 系统外,一些高校、科研部门也在开发免费的通用 MPI 系统,其中比较著名的有:

- MPICH(http://www.mcs.anl.gov/mpi/mpich)。
- LAM MPI(http://www.lam-mpi.org/)。

MPI 的发展历程如下:

- 1992 年 4 月,组建了一个制定消息传递接口标准的工作组。
- 1992 年 10 月,初稿形成,主要定义了点对点通信接口。
- 1993 年 1 月,第一届 MPI 会议在 Dallas 举行。
- 1993 年 2 月,公布了 MPI-1 修定版本。
- 1993 年 11 月,MPI 的草稿和概述发表在 Supercomputing'93 的会议论文集中。
- 1994 年 5 月,MPI 标准正式发布。
- 1994 年 7 月,发布了 MPI 标准的勘误表。
- 1997 年,MPI 论坛发布了一个修订的标准,叫做 MPI-2,同时,原来的 MPI 更名为 MPI-1。

一个 MPI 程序的各个进程通过调用 MPI 函数进行通信,协同完成一项计算任务。在 MPI 的 C 语言接口中,所有函数名均采用 MPI_Xxxxx 的形式,如 MPI_Send、MPI_Type_commit 等,它们以 MPI_开始,以便与其他函数名相区别,前缀 MPI_之后的第一个字母大写,其余字母一律小写。MPI 的 FORTRAN 接口定义为一组 SUBROUTINE,名称与 C 语言接口函数相同,由于 FORTRAN 语言中不区分字母的大小写,因此 MPI 函数、变量的名称写成大写或小写均可。

MPI 的 C 语言接口函数通常返回一个整数值表示操作成功与否,返回值为 MPI_SUCCESS(0)表示操作成功,否则表示操作的错误码。FORTRAN 接口比相应的 C 语言接口函数多出一个整型参数,用于返回错误码。只有两个例外是 MPI_Wtime 和 MPI_Wtick,它们在 C 语言和 FORTRAN 中均采用函数的形式,返回值类型分别为 double(C)和 DOUBLE PRECISION(FORTRAN)。

一段典型的 C 语言编写的 MPI 程序架构如下所示:

```
/* 文件名:hello.c */
#include "mpi.h"
#include <studio.h>
int main(int argc, char ** argv)
{
    MPI_Init(&argc, &argv);         //并行部分开始
    printf("hello parallel world!\n");
    MPI_Finalize();                 //并行部分结束
}
```

用 C 语言编写的 MPI 程序中每个源文件必须包含 MPI 的 C 语言头文件 mpi.h,以

便得到 MPI 函数的原型说明及 MPI 预定义的常量和类型。需要注意的是源文件中包含头文件 mpi. h 时不要含路径,必要时可在编译时通过-I 选项指定 mpi. h 所在的路径,以方便程序在不同 MPI 系统间的移植。

MPI_Init 函数用于初始化 MPI 系统。在调用其他 MPI 函数前(除 MPI_Initialized 外)必须先调用该函数。在许多 MPI 系统中,第一个进程通过 MPI_Init 来启动其他进程,需要将命令行参数的地址(指针)&argc 和 &argv 传递给 MPI_Init。MPI 程序启动时一些初始参数是通过命令行传递给进程的,这些参数被添加在命令行参数表中,MPI_Init 通过它们得到 MPI 程序运行的相关信息,如需要启动的进程数、使用哪些节点以及进程间的通信端口等。因此,一个 MPI 程序如果需要处理命令行参数,最好在调用 MPI_Init 之后再进行处理,这样可以避免遇到 MPI 系统附加的额外参数。MPI_Finalize 函数用于退出 MPI 系统。调用 MPI_Finalize 之后不能再调用任何其他 MPI 函数。

如图 10-1 所示,当本段程序在一个具有 4 个节点的集群上执行时,该程序会分别运行在不同节点上,并得到 4 条输出语句"hello parallel world!"。

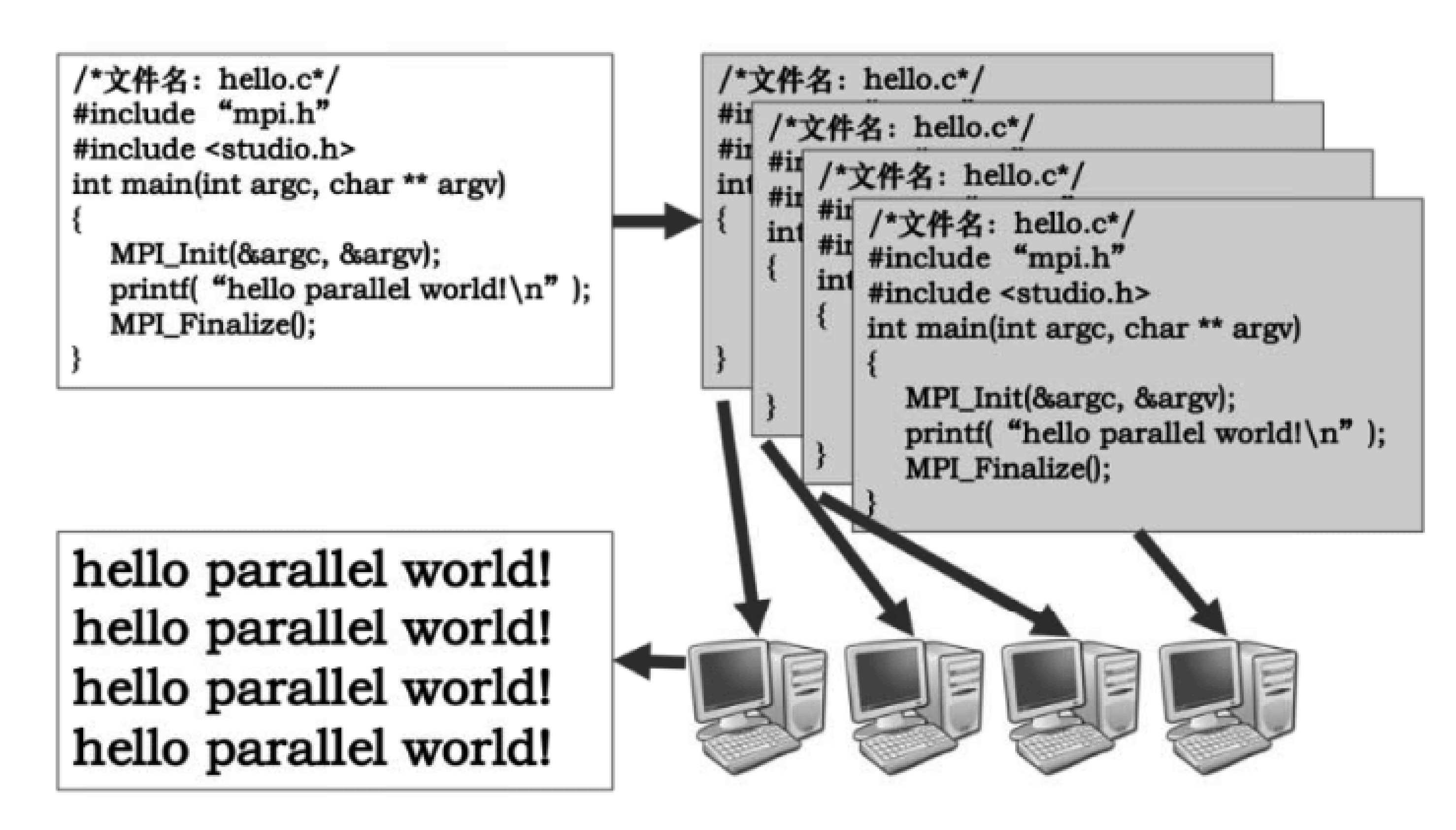


图 10-1 MPI 程序执行过程

2. 并行计算模型

在大数据背景下,随着以 Hadoop/Spark 为代表的大规模数据分布式并行计算模型的出现并获得巨大成功,工业界和学术界对于并行计算模型的关注越来越高。

1) 面向批处理的并行计算模型

具有代表性的面向批处理式分析的分布式并行计算模型有微软公司的 Dryad、谷歌公司的 MapReduce 等。MapReduce 已经成为学术界和工业界事实上的海量数据并行批量处理的标准。在第 8 章已经系统介绍了 MapReduce 计算框架,相比于传统的并行计算技术,MapReduce 具备了线性可扩展性、高可用性、易用性、容错性、负载平衡以及鲁棒性等大数据处理系统所必备的特性。用户在使用 MapReduce 并行编程模型时只需要关注与具体应用相关的高层处理逻辑,而将其余低层复杂的并行事务(如输入分布、任务划分与调度、任务间通信、容错处理以及负载平衡等)交与执行引擎完成。同时,配合用户自定义的输入输出流处理、任务调度、中间数据分区和排序等可编程接口,该模型在可伸缩性

和可编程性上达到了较好的平衡点。

2）面向流处理的并行计算模型

在大数据应用中，实时数据流应用是一类新型的用瞬态数据建模的数据密集型应用。其典型特征是数据价值具有时效性，即数据蕴含的价值会随着时间的流逝而降低。因此，低延迟是对数据流处理系统的一个基本要求。同时，数据以大量、连续、快速、时变的方式到达系统，需要处理的数据不可能全部存储在可随机访问的磁盘或内存中。近年来，随着社交网络的急速发展，大规模高扩展的流式计算模型成为业界研究的热点，具有代表性的有 Yahoo!的 S4、Facebook 的 Puma、谷歌的 MillWheel、Twitter 的 Storm 等，这些系统与企业自身的具体需求紧密结合，致力于解决实际的应用问题。相比于批处理计算模型，上述流式并行计算模型从流数据本身的特征出发，从底层架构上就与流数据处理高度耦合，虽然适用范围比较受局限，但是可以有效地将系统响应时间控制在毫秒级。流式并行计算模型的不足是在吞吐能力、负载平衡等方面尚有待进一步提高。

3）面向大图数据的并行计算模型

在大数据时代背景下，数十亿顶点级别大规模图的不断涌现以及云计算基础设施的持续完善，推动着图数据处理的研究重心由单机图算法的高度优化逐渐转向为分布式并行大图处理的优化。目前，大图数据处理存在两种典型的模式：一是采用通用的海量数据分布式并行计算框架 MapReduce 进行处理，二是采用完全面向图结构设计的专用大图计算框架。在分布内存架构下，目前具有代表性的大图并行计算模型有 Pregel、HAMA、Giraph、Distributed GraphLab 以及 Trinity 等。

4）基于内存的并行计算模型

受限于廉价 PC 构建的运行环境以及面向领域高度优化的系统架构，上述并行计算模型难以有效应对新型实时型应用对于实时、即席、交互式分析的复杂业务诉求。在基于内存的面向大数据的分布式并行计算模型研究工作开展之前，工业界和学术界在基于内存的数据管理技术，特别是主存数据库领域已经累积了大量的研究成果和经验。近年来，在新型实时应用的驱动下，以最短响应时间为设计目标的面向内存设计的编程模型及其系统也在不断涌现，为即席实时可交互的分析提供了多样化的选择。其中具有代表性的是 UCBerkeley 的基于内存的分布式并行处理框架 Spark，其利用内存计算避免了高延迟的磁盘物化，有效保证了处理的实时性并提供了交互式的迭代分析能力。读者在第 8 章系统学习了 Spark 的相关理论基础和编程方法，Spark 提供的最主要的抽象即弹性分布式数据集(RDD)，为了提供操作的便捷性，Spark 框架还提供了和 Hive 类似的类 SQL 命令接口 Shark。同时，基于 Spark 的内存计算分析生态系统已经越来越丰富，如处理流数据的 Spark Streaming，用于大图计算的 GraphX 等。

10.1.3 并行算法设计的策略和技术

并行算法的设计在处理数值型和非数值型数据类型时有一定的相似性，一般有 3 种设计策略：

(1) 串行算法的直接并行化。就是充分发掘和利用现有串行算法中的并行性，直接将串行算法改造为并行算法，这是数值并行算法中最常用的设计策略。

(2) 从问题描述和计算原理开始重新设计。从问题本身描述出发,不考虑相应的串行算法,设计一个全新的并行算法。

(3) 借用已有的算法求解新问题。借鉴别处的已有算法,解决新发现的问题,可能会取得令人满意的结果。

采用PCAM(Partitioning-Communication-Agglomeration-Mapping)设计方法学设计并行算法分为4个阶段:划分、通信、组合、映射。它反映了并行算法设计的自然过程,首先尽量开拓算法的并发性,满足算法的可扩展性,然后着重优化算法的通信成本和全局执行时间,同时通过必要的整个过程的反复回溯,以期最终达到一个满意的设计,具体过程如图10-2所示。

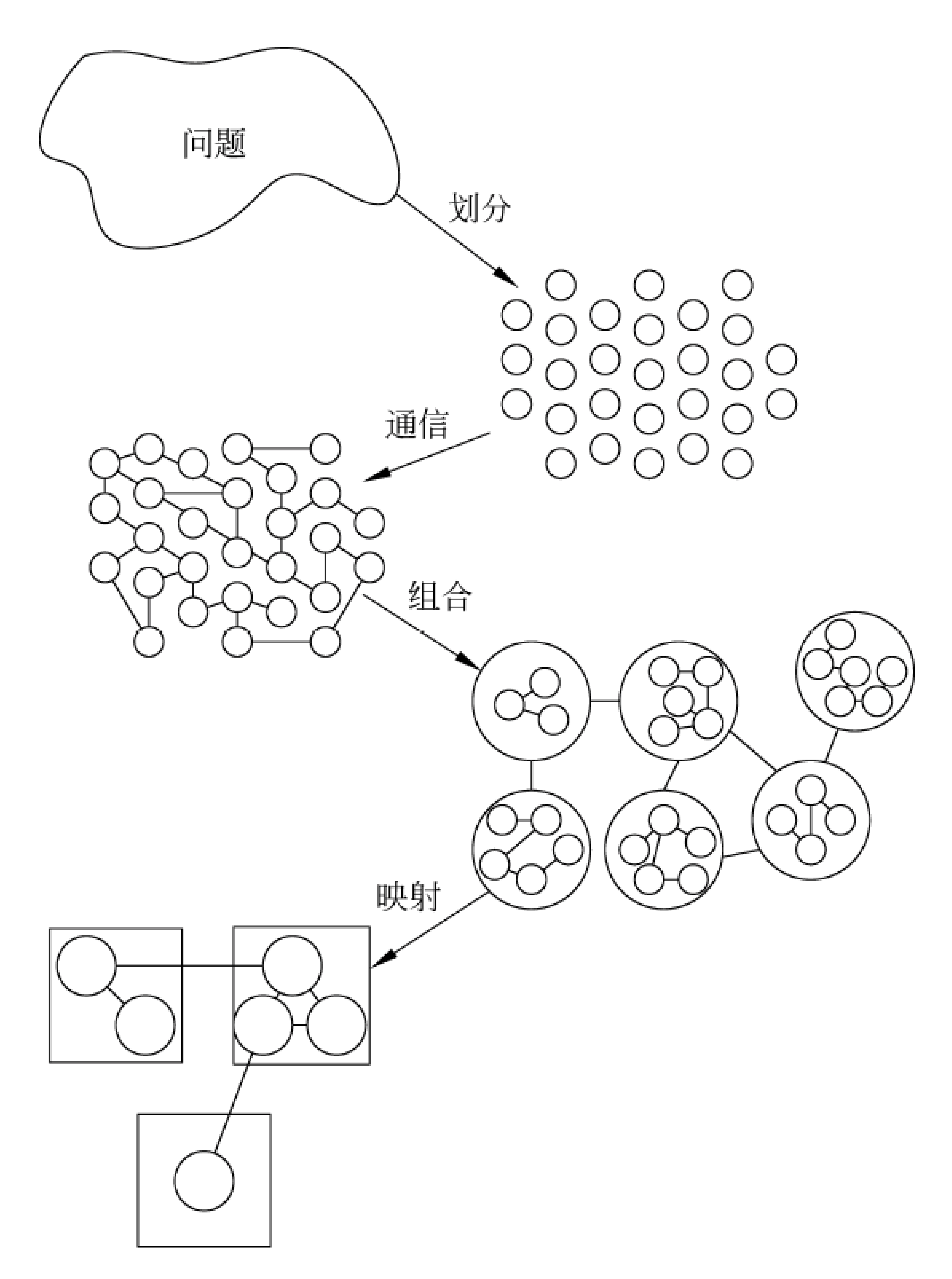


图10-2 PCAM求解问题过程

(1) 划分(Partitioning)。分解成小的任务,开拓并发性。主要使用域分解和功能分解方法将计算问题分解成若干个可部分执行或完全执行的子任务。

- 域分解。又称数据划分,是将问题相关的大块数据尽可能分割成大小均匀的小数据片,并把计算关联到它所操作的数据上。典型代表包括分治法、偏微分方程数值求解的区域分解法、方程组迭代求解中的红/黑着色法等。
- 功能分解。又称计算划分,是以被执行的计算作为划分的对象,而不是计算所需的数据。典型代表包括流水(脉动)方法等。

(2) 通信(Communication)。确定诸任务间的数据交换,监测划分的合理性。计算问

题划分得到诸任务后常常存在彼此的数据依赖，而不能完全独立地执行，因而子任务间数据依赖关系的差别便产生了不同的通信模式，包括全部/全局通信、结构化/非结构化通信、静态/动态通信、同步/异步通信等。

（3）组合（Agglomeration）。依据任务的局部性，组合成更大的任务。其主要目的是通过合并小尺寸的任务来减少任务数，以提高效率和减少通信成本。

- 表面-容积效应。即每个子任务有较少的通信对象时，则增大划分粒度能减少通信次数，并同时减少总通信量。
- 重复计算。即采用不必要的多余计算来获得相关数据以减少通信要求或执行时间。

（4）映射（Mapping）。将每个任务分配到处理器上，提高算法的性能，减少算法的总执行时间。主要策略有两种：一是将能并发执行的任务映射到不同处理器上以增强并行度，二是将需要频繁通信的任务映射到同一处理器以提高局部性。

- 负载平衡。可采用递归对剖法、局部算法、概率方法、循环映射等。其中递归对剖法又能分化出递归坐标对剖、非平衡递归对剖、递归图剖分等方法。循环映射能衍生出反射映射、块循环映射等方法。
- 任务调度。一类是经理/雇员模式，在大规模问题中常采用层次经理/雇员模式以避免经理成为瓶颈；另一类是非集中模式，并且任务调度算法需要执行结束检测来检测操作是否结束。

10.2 典型数据挖掘算法并行化案例

本节给出并行算法相关的基本概念及其应用术语，为读者学习并行计算及其编程实现建立基础。首先介绍并行算法的概念，然后介绍并行计算的模型，最后介绍并行算法设计的策略和技术。

聚类分析作为一种无监督学习，一般用来对数据对象按照其特征属性进行分组，经常被应用在欺诈检测、图像分析等领域。其中，k-means 是最有名并且最经常使用的聚类算法了，其原理比较容易理解，并且聚类效果良好，得到了广泛的应用。本节以 k-means 算法为例，讲解 MapReduce 环境下并行算法设计的方法和过程。

10.2.1 MR k-means 算法分析

首先，回顾一下 k-means 算法的基本步骤：

（1）从数据集 x 中随机选择一个点作为第一个初始点。

（2）计算数据集中所有点与当前中心点的距离 $D(x)$。

（3）选择下一个中心点，使得 $P(x)=\dfrac{D(x)^2}{\sum_{x_e} D(x)^2}$ 最大。

（4）重复（1）、（2）步过程，直到 k 个初始点选择完成。

假设样本数据有 n 个，预期生成 k 个 cluster，则 k-means 算法 t 次迭代过程的时间复杂度为 $O(nkt)$，需要计算 ntk 次相似度，那么在 MapReduce 计算框架下，如果能够将各个

点到中心点的相似度计算工作分摊到不同的计算机上并行地计算，是不是能够减少计算时间呢？通过思考可以发现，在 *k*-means 中处理每一个数据点时，每个聚簇的中心点信息是始终需要用到的，而其他点的信息只需要在比对时读入当前点的信息即可。所以，如果涉及全局信息，只需要知道关于各个聚簇的信息即可。因此，可以尝试从以下出发点进行 MapReduce *k*-means 算法改造：

- 将所有的数据分布到不同的节点上，每个节点只对自己的数据进行计算。
- 每个节点能够读取上一次迭代生成的聚簇中心，并判断自己的各个数据点应该属于哪一个聚簇。
- 每个节点在每次迭代中根据自己的数据点计算出相关结果。
- 综合每个节点计算出的相关数据，计算出最终的实际聚类结果。

总结一下，需要关注的参数就是迭代次数 k、聚类 ID、聚类中心和属于该聚类中心的数据点总数。接下来，具体分析算法的执行过程，并查看这些参数的变化情况。假定有如表 10-2 所示的数据集，可看做二维坐标系中的 4 个点{$A(1,1)$，$B(2,3)$，$C(3,4)$，$D(5,5)$}，迭代次数 $k=2$，聚类个数 $n=2$，其执行过程如下。

表 10-2 待聚类数据集

对象	属性 1(*X*)	属性 2(*Y*)	对象	属性 1(*X*)	属性 2(*Y*)
A	1	1	*C*	3	4
B	2	3	*D*	5	5

假定将所有数据分布到两个节点 node-0 和 node-1 上，即 node-0：$A(1,1)$和 $C(3,4)$，node-1 ：$B(2,3)$和 $D(5,5)$，随机选取 $A(1,1)$作为 cluster-0 的中心，$C(3,4)$作为 cluster-1 的中心。迭代开始前的聚簇情况如表 10-3 所示。

表 10-3 初始划分

聚簇 ID	聚簇中心	被分配的点数
cluster-0	$A(1,1)$	0
cluster-1	$C(3,4)$	0

Map 阶段：计算各个数据点到各个 cluster 中心的距离。通过计算可知，点 A 和 C 到自身的距离分别是 0，是最近的，因此应将其分配到对应的聚簇中。分别分析点 B、D 与 A、C 的距离，发现 B 和 D 均距离点 C 更近一些，那么应该把 B、D 暂时归入 C 所在的聚簇，并标记为 cluster-1。

表 10-4 Map 阶段初次距离计算

数据点	到各个聚簇的距离比较		分配的聚簇
	cluster-0	cluster-1	
$A(1,1)$	0，近		Cluster-0
$C(3,4)$		0，近	Cluster-1

续表

数据点	到各个聚簇的距离比较		分配的聚簇
	cluster-0	cluster-1	
$B(2,2)$	$\sqrt{5}$	$\sqrt{2}$，近	Cluster-1
$D(5,5)$	$4\sqrt{2}$	$\sqrt{5}$，近	Cluster-1

这时在每个节点上的数据输出可按照下表进行标记，接下来在 Combine 阶段，Map 的输出也就是 Combiner 的输入。

Node-0 输出：	Node-1 输出：
<cluster-0，A(1,1)>	<cluster-1，B(2,3)>
<cluster-1，C(3,4)>	<cluster-1，D(5,5)>

经过计算，Combiner 的输出采用如下的键值对格式：

- 键—聚簇 ID。
- 值—[包含的点数，均值]。

Node-0 输出：	Node-1 输出：
<cluster-0，[1,(1,1)]>	<cluster-1，[2,(3.5,4)]>
<cluster-1，[1,(3,4)]>	

接下来进入 Reduce 阶段，由于 Map 阶段输出的键是聚簇 ID，所以每个聚簇的全部数据将被发送给同一个 Reducer，包括聚簇 ID、该聚簇的数据点的均值以及对应于该均值的数据点的个数。两个 Reducer 分别收到

Reducer-0：	Reducer-1：
<cluster-0，[1,(1,1)]>	<cluster-1,[1,(3,4)]
	<cluster-1,[2,(3.5,4)]

经计算可知，得到两个聚簇 cluster-0 和 cluster-1，当满足终止条件时，即可停止迭代，输出 k 个聚类。在 MR k-means 中，终止条件的设置与原 k-means 可保持一致。例如，设定迭代次数、均方差的变化(非充分条件)、指定的点固定地属于某个聚类等。

10.2.2 Mahout 聚类算法案例

Apache Mahout 是 Apache Software Foundation (ASF)开发的一个全新的开源项目，其主要目标是建立一个可靠、文档翔实、可伸缩的项目，在其中实现一些常见的机器学习算法，供开发人员在 Apache 许可下免费使用。Mahout 最大的优点就是基于 Hadoop 实现，把以前运行于单机上的经典算法转换到 MapReduce 计算框架下，大大提升了算法可处理的数据量和处理性能。在 Mahout 上实现的机器学习算法如表 10-5 所示。

表 10-5 在 Mahout 上实现的机器学习算法

算法分类	算法名称	中文名称
分类算法	Logistic Regression	逻辑回归
	Bayesian	贝叶斯
	SVM	支持向量机
	Perceptron	感知器
	Neural Network	神经网络
	Random Forests	随机森林
	Restricted Boltzmann Machines	有限波尔兹曼机
聚类算法	Canopy Clustering	Canopy 聚类
	k-means Clustering	*k* 均值聚类
	Fuzzy *k*-means	模糊 *k* 均值
	Expectation Maximization	EM 聚类(期望最大化聚类)
	Mean Shift Clustering	均值漂移聚类
	Hierarchical Clustering	层次聚类
	Dirichlet Process Clustering	狄里克雷过程聚类
	Latent Dirichlet Allocation	LDA 聚类
	Spectral Clustering	谱聚类
关联规则挖掘	Parallel FP Growth Algorithm	并行 FP Growth 算法
回归	Locally Weighted Linear Regression	局部加权线性回归
降维/维约简	Singular Value Decomposition	奇异值分解
	Principal Components Analysis	主成分分析
	Independent Component Analysis	独立成分分析
	Gaussian Discriminative Analysis	高斯判别分析
进化算法	并行化的 Watchmaker 框架	
推荐/协同过滤	Non-Distributed Recommenders	Taste(UserCF, ItemCF, SlopeOne)
	Distributed Recommenders	ItemCF
向量相似度计算	Row Similarity Job	计算列间相似度
	Vector Distance Job	计算向量间距离
非 MapReduce 算法	Hidden Markov Models	隐马尔可夫模型
集合方法扩展	Collections	扩展了 Java 的 Collections 类

Mahout 的安装和使用比较简单,直接解压使用或在相应的函数调用时引入即可。安装过程简单描述如下。

使用下面的命令解压 Mahout 安装包：

```
cd /Mahout 压缩包所在目录/
mv mahout-distribution-0.9.tar.gz ~/
cd
tar -zxvf ~/mahout-distribution-0.9.tar.gz
cd mahout-distribution-0.9
```

执行 ls -l 命令，会看到如图 10-3 所示的内容，其中展示了 mahout-distribution-0.9 目录下包含的文件。

```
[zhangxu@master ~]$ cd mahout-distribution-0.9
[zhangxu@master mahout-distribution-0.9]$ ls -l
total 39880
drwxrwxr-x. 2 zhangxu zhangxu     4096 Aug  2 20:26 bin
drwxr-xr-x. 2 zhangxu zhangxu     4096 Jan 28  2014 conf
drwxrwxr-x. 6 zhangxu zhangxu     4096 Aug  2 20:26 docs
drwxrwxr-x. 4 zhangxu zhangxu     4096 Aug  2 20:26 examples
drwxrwxr-x. 3 zhangxu zhangxu     4096 Aug  2 20:26 lib
-rw-r--r--. 1 zhangxu zhangxu    39588 Jan 28  2014 LICENSE.txt
-rw-r--r--. 1 zhangxu zhangxu  1469563 Jan 28  2014 mahout-core-0.9.jar
-rw-r--r--. 1 zhangxu zhangxu 12831506 Jan 28  2014 mahout-core-0.9-job.jar
-rw-r--r--. 1 zhangxu zhangxu   235053 Jan 28  2014 mahout-examples-0.9.jar
-rw-r--r--. 1 zhangxu zhangxu 24178445 Jan 28  2014 mahout-examples-0.9-job.jar
-rw-r--r--. 1 zhangxu zhangxu   432668 Jan 28  2014 mahout-integration-0.9.jar
-rw-r--r--. 1 zhangxu zhangxu  1612814 Jan 28  2014 mahout-math-0.9.jar
-rw-r--r--. 1 zhangxu zhangxu     1888 Jan 28  2014 NOTICE.txt
-rw-r--r--. 1 zhangxu zhangxu     1212 Jan 28  2014 README.txt
[zhangxu@master mahout-distribution-0.9]$
```

图 10-3　Mahout 主目录结构

解压安装后，可启动并验证 Mahout，执行的操作命令如下：

```
cd /Mahout 解压安装目录/
bin/mahout
```

执行命令后看到类似图 10-4 的输出，则表示安装成功。

从 Mahout 源码可以看到，在进行 k-means 聚类时，执行以下 4 个步骤：

(1) 数据预处理，整理规范化数据。

(2) 从上述数据中随机选择若干个数据当作聚簇的中心。

(3) 迭代计算，调整形心。

(4) 把数据分给各个聚簇。

其中，前两步就是标准 k-means 聚类算法的准备工作，后面的步骤和 10.2.1 节分析的 MR k-means 案例思路一致，其主要流程可以从 org.apache.mahout.clustering.syntheticcontrol.kmeans.Job#run()方法里看出。

```
Public static void run(Configuration conf, Path input, Path output,DistanceMeasure
measure,int k,double convergenceDelta,int maxIterations) throws Exception {
  //synthetic_control.data 存储的文本格式,以 KV 形式存入 output/data 目录。
Path directoryContainingConvertedInput=new Path(output DIRECTORY_CONTAINING_
CONVERTED_INPUT);
```

```
[zhangxu@master ~]$ cd /home/zhangxu/mahout-distribution-0.9
[zhangxu@master mahout-distribution-0.9]$ bin/mahout
Running on hadoop, using /home/zhangxu/hadoop-2.5.2/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/zhangxu/mahout-distribution-0.9/mahout-examples-0.9-job.jar
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zhangxu/hadoop-2.5.2/share/hadoop/common/lib/slf4
j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zhangxu/hbase-0.98.9-hadoop2/lib/slf4j-log4j12-1.
6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
An example program must be given as the first argument.
Valid program names are:
  arff.vector: : Generate Vectors from an ARFF file or directory
  baumwelch: : Baum-Welch algorithm for unsupervised HMM training
  canopy: : Canopy clustering
  cat: : Print a file or resource as the logistic regression models would see it
  cleansvd: : Cleanup and verification of SVD output
  clusterdump: : Dump cluster output to text
  clusterpp: : Groups Clustering Output In Clusters
  cmdump: : Dump confusion matrix in HTML or text formats
  concatmatrices: : Concatenates 2 matrices of same cardinality into a single matrix
  cvb: : LDA via Collapsed Variation Bayes (0th deriv. approx)
  cvb0_local: : LDA via Collapsed Variation Bayes, in memory locally.
  evaluateFactorization: : compute RMSE and MAE of a rating matrix factorization against
probes
  fkmeans: : Fuzzy K-means clustering
  hmmpredict: : Generate random sequence of observations by given HMM
  itemsimilarity: : Compute the item-item-similarities for item-based collaborative filte
ring
  kmeans: : K-means clustering
  lucene.vector: : Generate Vectors from a Lucene index
  lucene2seq: : Generate Text SequenceFiles from a Lucene index
  matrixdump: : Dump matrix in CSV format
```

图 10-4 Mahout 测试安装成功的运行界面

```
log.info("Preparing Input");
InputDriver.runJob(input, directoryContainingConvertedInput,"org.apache.
mahout.math.RandomAccessSparseVector");
  //随机产生几个聚簇,存入 output/clusters-0/part-randomSeed 文件
log.info("Running random seed to get initial clusters");
Path clusters=new Path(output,Cluster.INITIAL_CLUSTERS_DIR);
clusters=RandomSeedGenerator.buildRandom(conf, directoryContaining-
ConvertedInput, clusters, k, measure);
  //进行聚类迭代运算,为每一个簇重新选出聚簇中心
log.info("Running KMeans");
KMeansDriver.run(conf, directoryContainingConvertedInput, clusters, output,
measure, convergenceDelta,maxIterations,true, 0.0,false);
//根据上面选出的中心,把 output/data 中的记录都分配给各个聚簇,最后输出运算结果
ClusterDumper clusterDumper=new ClusterDumper(new Path(output, "clusters-*-final"),new
Path(output,"clusteredPoints"));
clusterDumper.printClusters(null);
}
```

在上面的代码中:

- 参数 input 指定待聚类的所有数据点,clusters 指定初始聚类中心。
- 参数 output 指定聚类结果的输出路径。

- clusters-N 目录保存根据原数据点和上一次迭代(或初始聚类)的聚类中心计算得到的本次迭代的聚类中心,该过程由 org. apache. mahout. clustering. kmeans 下的 KMeansMapper\KMeansCombiner\KMeansReducer\KMeansDriver 实现。
- KMeansMapper 在初始化 Mapper 时读入上一次迭代产生或初始的全部聚类中心,然后通过 Map 方法对输入的每个点计算距离其最近的类,并加入其中。输出 key 为该点所属聚类 ID,value 为 KMeansInfo 实例,包含点的个数和各分量的累加和。
- KMeansCombiner 在本地累加 KMeansMapper 输出的同一聚类 ID 下的点个数和各分量的和。
- KMeansReducer 累加同一聚类 ID 下的点个数和各分量的和,求本次迭代的聚类中心,并判断该聚类是否已收敛,最后输出各聚类中心和其是否收敛标记。
- KMeansDriver 在每轮迭代后读取其 clusters-N 目录下的所有聚类,若所有聚类已收敛,则整个 k-means 聚类过程就收敛了。

回想并对比一下在 10.2.1 节讲解的 MR k-means 实现原理,可以发现 Mahout 很好地实现了这样一个算法,不需要再自行编程实现,只需要按照参数要求调用即可。当然,如果需要对算法进行优化,还是需要考虑自行实现该算法。

接下来分析如何调用 Mahout 提供的 k-means 算法实现对控制时序数据的聚类。首先,下载控制时序数据(http://archive. ics. uci. edu/ml/databases/synthetic_control/synthetic_control. data),如图 10-5 所示,每幅小图代表一类数据:趋势向下(A)、周期(B)、正常(C)、向上偏移(D)、趋势向上(E)、向下偏移(F)。通过调用 Mahout 的 k-means 聚类方法分析该数据的过程如下:

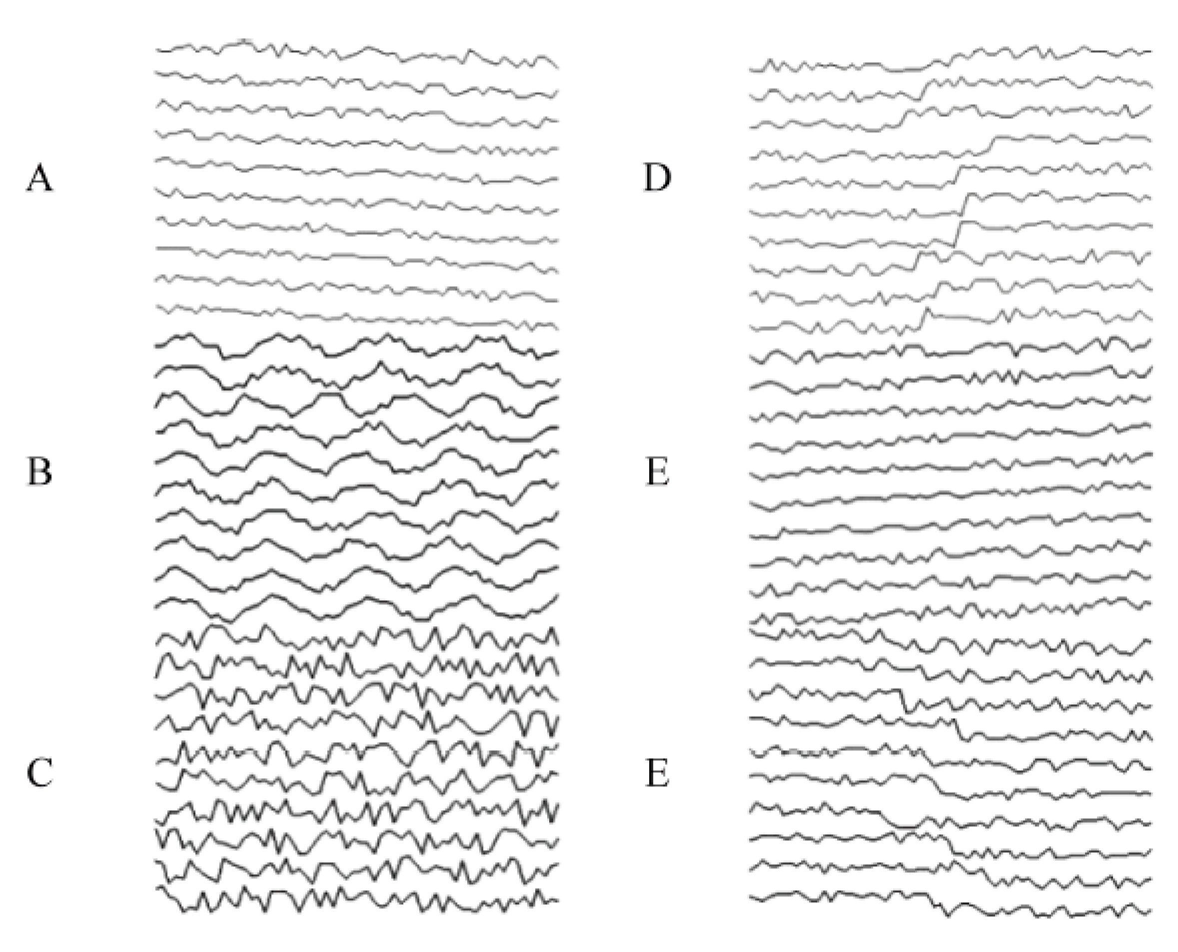


图 10-5 控制时序聚类数据

(1) 上传实验数据到 HDFS 文件系统:

```
hadoop fs -put synthetic_control.data /user/root/testdata
```

（2）运行聚类程序：

```
hadoop jar /home/zhangxu/mahout-distribution-0.9/mahout-examples-0.9-job.jar org.
apache.mahout.clustering.syntheticcontrol.kmeans.Job
```

（3）查看聚类结果，如图 10-6 所示。

```
-sh-3.2# hadoop fs -ls /user/root/output        K-means聚类后结果
Found 15 items
-rw-r--r--   1 root supergroup        194 2014-04-26 18:31 /user/root/output/_policy
drwxr-xr-x   - root supergroup          0 2014-04-26 18:32 /user/root/output/clusteredPoints
drwxr-xr-x   - root supergroup          0 2014-04-26 18:24 /user/root/output/clusters-0
drwxr-xr-x   - root supergroup          0 2014-04-26 18:25 /user/root/output/clusters-1
drwxr-xr-x   - root supergroup          0 2014-04-26 18:31 /user/root/output/clusters-10-final
drwxr-xr-x   - root supergroup          0 2014-04-26 18:26 /user/root/output/clusters-2
drwxr-xr-x   - root supergroup          0 2014-04-26 18:27 /user/root/output/clusters-3
drwxr-xr-x   - root supergroup          0 2014-04-26 18:27 /user/root/output/clusters-4
drwxr-xr-x   - root supergroup          0 2014-04-26 18:28 /user/root/output/clusters-5
drwxr-xr-x   - root supergroup          0 2014-04-26 18:29 /user/root/output/clusters-6
drwxr-xr-x   - root supergroup          0 2014-04-26 18:29 /user/root/output/clusters-7
drwxr-xr-x   - root supergroup          0 2014-04-26 18:30 /user/root/output/clusters-8
drwxr-xr-x   - root supergroup          0 2014-04-26 18:31 /user/root/output/clusters-9
drwxr-xr-x   - root supergroup          0 2014-04-26 18:24 /user/root/output/data
```

图 10-6　时序控制数据 *k*-means 聚类结果目录

10.2.3　Spark MLlib 聚类算法案例

通过前面的学习，可以了解到 *k*-means 算法和诸多机器学习算法一样，也是一个迭代式的算法，那么能否在 Spark 平台下实现呢？值得高兴的是，Spark 平台下提供了一个很好的机器学习库——MLlib，类似于 MapReduce 下的 Mahout。Spark MLlib *k*-means 算法在初始聚类点的选择上遵循一个基本原则：初始聚类中心点相互之间的距离应该尽可能远。MLlib 提供的 *k*-means 聚类包含的关键参数如图 10-7 所示。

```
class KMeans private (
    private var k: Int,
    private var maxIterations: Int,
    private var runs: Int,
    private var initializationMode: String,
    private var initializationSteps: Int,
    private var epsilon: Double,
    private var seed: Long) extends Serializable with Logging
```

图 10-7　MLlib 机器学习库的 *k*-means 聚类参数

其中：

- k 为期望的聚类个数。
- maxInterations 为单次运行最大的迭代次数。
- runs 为算法运行的次数。*k*-means 算法不保证能返回全局最优的聚类结果，所以在目标数据集上多次执行 *k*-means 算法，有助于返回最佳聚类结果。
- initializationMode 为初始聚类中心点的选择方式。
- initializationSteps 为 *k*-means 方法中的步数。
- epsilon 表示 *k*-means 算法迭代收敛的阈值。
- seed 表示集群初始化时的随机种子。

通常，在应用时都会先调用 KMeans.train 方法对数据集进行聚类训练，这个方法会

返回 KMeansModel 类实例，也可以使用 KMeansModel. predict 方法对新的数据点进行所属聚类的预测。使用方法如下面的代码所示，该示例程序接受 5 个输入参数，分别是训练数据集文件路径、测试数据集文件路径、聚类个数、迭代次数、运行次数。

```
/*
 * import 部分省略
 */
object KMeansClustering {
def main (args: Array[String]) {
if (args.length < 5) {
    //5 个输入参数：训练数据集文件路径、测试数据集文件路径、聚类个数、迭代次数、运行次数
    println("Usage: KMeansClustering trainingDataFilePath testDataFilePath numClusters
    numIterations runTimes")
  sys.exit(1)
}
//初始化 SparkConf 配置
val conf=new SparkConf().setAppName("Spark MLlib K-Means Clustering")
val sc=new SparkContext(conf)
//5 个参数分别解析读入
val rawTrainingData=sc.textFile(args(0))
val parsedTrainingData=rawTrainingData.filter(!isColumnNameLine(_)).
map(line=>{
    Vectors.dense(line.split("\t").map(_.trim).filter(!"".equals(_)).
map(_.toDouble))
}).cache()
val numClusters=args(2).toInt
val numIterations=args(3).toInt
val runTimes=args(4).toInt
var clusterIndex: Int=0
//按照指定参数进行聚类训练并返回 KMeansModel 类实例
val clusters: KMeansModel=KMeans.train(parsedTrainingData, numClusters,
numIterations,runTimes)
  //输出聚类信息
      println("Cluster Number: " +clusters.clusterCenters.length)
      println("Cluster Centers Information Overview: ")
  clusters.clusterCenters.foreach(
      x=>{
      println("Center Point of Cluster " +clusterIndex +": ")
      println(x)
  clusterIndex +=1
  })
    //开始检测每个测试数据所属的簇
```

```
val rawTestData=sc.textFile(args(1))
val parsedTestData=rawTestData.map(line=>
    {
    Vectors.dense(line.split("\t").map(_.trim).filter(!"".equals(_)).
    map(_.toDouble))
    })
parsedTestData.collect().foreach(testDataLine=>{
val predictedClusterIndex:
    Int=clusters.predict(testDataLine)
    println("The data " +testDataLine.toString +" belongs to cluster " +
    predictedClusterIndex)
})
    println("Spark MLlib k-means clustering test finished.")
}
private def
isColumnNameLine(line: String): Boolean={
if (line !=null&& line.contains("Channel"))
  true
else
  false
}
```

前面提到 k 的选择是 k-means 算法的关键，Spark MLlib 在 KMeansModel 类里提供了 computeCost 方法，该方法通过计算所有数据点到其最近的中心点的平方和来评估聚类的效果。一般来说，同样的迭代次数和算法执行次数，这个值越小，代表聚类的效果越好。但是在实际情况下，因要考虑到聚类结果的可解释性，不能一味地选择使 computeCost 结果值最小的 k。

10.3 大数据分析应用案例

通过前面章节的学习，相信读者已经掌握了大数据分析算法在分布式并行环境下的基本思想和编程方法，特别是对 MapReduce 和 Spark 环境开发大数据算法有了初步的认识。接下来，本节讲解如何把这些思想和方法应用到实际案例的分析中。

10.3.1 搜索引擎日志数据分析

本案例使用某搜索引擎在 2012 年采集的部分用户搜索数据，已做脱敏处理。样例数据如表 10-6 所示。数据字段为用户搜索时间、用户 ID、查询词、该 URL 在返回结果中的排名、用户点击的顺序号、用户点击的 URL。其中，用户 ID 根据用户使用浏览器访问搜索引擎时的 Cookie 信息自动赋值，即同一次使用浏览器输入的不同查询对应同一个用户 ID。

表 10-6　搜索日志数据样例

NO	UTIME	UID	KEYWORD	R	CR	CURL
1	20120405000008	6961d0c97fe93701fc9c0d861d096cd9	重庆邮电大学	1	1	http://www.cqupt.edu.cn/
2	20120405000009	96994a0480e7e1edcaef67b20d8816b7	伟大导演	1	1	http://movie.douban.com/review/1128960/
3	20120405000009	698956eb07815439fe5f46e9a4503997	youku	1	1	http://www.youku.com/
4	20120405000010	f4ba3f337efb1cc469fcd0b34feff9fb	推荐待机时间长的手机	1	1	http://mobile.zol.com.cn/148/1487938.html
5	20120405000011	7c54c43f3a8a0af0951c26d94a57d6c8	百度一下 你就知道	1	1	http://www.baidu.com/
6	20120405000017	e767b76990f9232e525d5014d802fd29	中国移动网上营业厅	1	1	http://10086.cn/service/
7	20120405000027	4a6f0d5cc0bcf16e32e74ae49663b60d	baidu	2	1	http://site.baidu.com/
8	20120405000055	6c1a44f96478f31cd310b394b24b680f	支付宝	1	1	http://www.alipay.com/
9	20120405000058	6b276bd438cc5b0de1afdd708fc772c8	联通网上营业厅	1	1	http://www.10010.com/
10	20120405000066	bc490faba3016ece5863fce8a53cf130	犀牛科创	4	2	http://www.rhinoz.net

假设管理员希望能够统计以重庆邮电大学或 CQUPT 作为关键词进行搜索的记录数。回想一下 MapReduce 编程的基本思想，采用类似 wordcount 的基本方法统计某关键词被搜索的次数，需要做的主要是主类、MAP 类和 REDUCE 类的代码编写。在主类中，需要设置的几个关键部分包括指定 Job 名称、主类入口、Mapper 类入口和 Reducer 类入口。

main 函数调用 Jobconf 类来对 MapReduce Job 进行初始化，然后调用 setJobName() 方法命名这个 Job。对 Job 进行合理的命名有助于更快地找到 Job，以便在 JobTracker 和 Tasktracker 的页面中对其进行监视。接着设置 Job 输出结果<key，value>的中 key 和 value 数据类型，因为结果是<单词，个数>，所以 key 设置为 Text 类型，相当于 Java 中 String 类型。Value 设置为 IntWritable，相当于 Java 中的 int 类型。接下来设置 Job 处理的 Map(拆分)、Combiner(中间结果合并)以及 Reduce(合并)的相关处理类：

```
conf.setMapperClass(Map.class);
conf.setCombinerClass(Reduce.class);
conf.setReducerClass(Reduce.class);
```

需要说明的是，Combiner 可以有效降低数据传输产生的压力，在下面的示例代码中未使用，可在练习时自行实现。

```
/*
 * import 部分省略
 */
public class WebSearchMain {
    public static void main(String[] args) throws Exception {
        Configuration conf=new Configuration();
        String[] otherArgs=new GenericOptionsParser(conf, args).
                getRemainingArgs();
        if (otherArgs.length <2) {
            System.err.println("Usage: wordcount <in>  [<in> ...] <out> ");
            System.exit(2);
        }
        //设置 Job 名称
        Job job=new Job(conf, "WebSearch KeyWord Count Test");
        //指定 Jar 包的主类入口
        job.setJarByClass(WebSearchMain.class);
        //指定 Mapper 类入口
        job.setMapperClass(SplitMapper.class);
        //指定 Reducer 类入口
        job.setReducerClass(SumReducer.class);
        //输入输出配置
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        for (int i=0; i <otherArgs.length -1; ++i) {
            FileInputFormat.addInputPath(job,new Path(otherArgs[i]));
        }
        FileOutputFormat.setOutputPath(job,new Path(otherArgs[otherArgs.
            length -1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

SplitMapper 继承 org.apache.hadoop.mapreduce 包中的 Mapper 类，并重写其 map 方法。可以发现 map 方法中 value 值存储的是文本文件中的一行(以回车符为行结束标记)，而键值为该行的首字母相对于文本文件的首地址的偏移量。然后 StringTokenizer 类将每一行拆分成为一个个的单词，并将<word，1>作为 map 方法的结果输出，其余的工作都交由 MapReduce 框架处理。对于关键词的统计，仅需增加一个过滤条件，查找输入数组中 arr[2]字段包含待统计关键词的数据条目。

```
/*
 * import 部分省略
 */
public class SplitMapper extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one=new IntWritable(1);
```

```
        private Text word=new Text("sum");
        public void map(Object key, Text value, Context context)
                throws IOException, InterruptedException {
            String[] arr=value.toString().split("\t");
            //统计网页搜索日志中关键字字段包含"重庆邮电大学"或 CQUPT 的记录数
            if (arr[2].indexOf("CQUPT") >=0 || arr[2].indexOf("重庆邮电大学") >=0){
                context.write(word, one);
            }
        }
    }
```

SumReduce 过程需要继承 org.apache.hadoop.mapreduce 包中的 Reducer 类，并重写其 reduce 方法。Map 过程输出<key，values>中 key 为单个单词，而 values 是对应单词的计数值所组成的列表，Map 的输出就是 Reduce 的输入，所以 reduce 方法只要遍历 values 并求和，即可得到某个搜索词的总次数，也就是该搜索记录的总次数。

```
/*
 * import 部分省略
 */
public class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable>{
    private IntWritable result=new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
        int sum=0;
        for (IntWritable val : values) {
            sum +=val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

10.3.2　出租车轨迹数据分析

本案例使用某城市的出租车数据，包括时间信息、空间信息、上下乘客等信息，数据格式为 csv 文件，数据总条数超过 1500 万条。字段如下：

medallion：UUID。

hack_license：UUID。

store_and_fwd_flag：是否四驱。

pickup_datatime：客人上车时间。

dropoff_datatime：客人下车时间。

passenger_count：载客数量。

trip_time_in_secs：载客时间。

trip_distance：载客距离。

pickup_longitude：客人上车经度。

pickup_latitude：客人上车纬度。

dropoff_longitude：客人下车经度。

dropoff_latitude：客人下车纬度。

在本案例中，主要采用Hive数据表的形式进行数据分析，以方便更多数据分析师理解该案例，仅需要一些基本的SQL知识就可以实现。Hive的库、表等数据实际是HDFS文件系统中的目录和文件，让开发者可以通过SQL语句，像操作关系数据库一样操作文件内容，比如执行查询、统计、插入等操作。Hive可以将SQL转化为MapReduce任务，使用户摆脱烦琐的MapReduce编程过程，整个编译过程分为6个阶段：

（1）Antlr定义SQL的语法规则，完成SQL词法、语法解析，将SQL转化为抽象语法树AST Tree。

（2）遍历AST Tree，抽象出查询的基本组成单元QueryBlock。

（3）遍历QueryBlock，翻译为执行操作树OperatorTree。

（4）逻辑层优化器进行OperatorTree变换，合并不必要的ReduceSinkOperator，减少shuffle数据量。

（5）遍历OperatorTree，翻译为MapReduce任务。

（6）物理层优化器进行MapReduce任务的变换，生成最终的执行计划。

在使用Hive进行数据查询分析时可采用shell方式，也可以利用HUE或Zeppelin等可视化工具。在本例分析时，使用了Hue可视化分析工具，如图10-8所示。Hue是一个开源的Apache Hadoop UI系统，最早是由Cloudera Desktop演化而来的，由Cloudera贡献给开源社区，它是基于Python Web框架Django实现的。通过使用Hue可以在浏览器端的Web控制台上与Hadoop集群进行交互来分析处理数据，例如操作HDFS上的数据，运行MapReduce Job等。

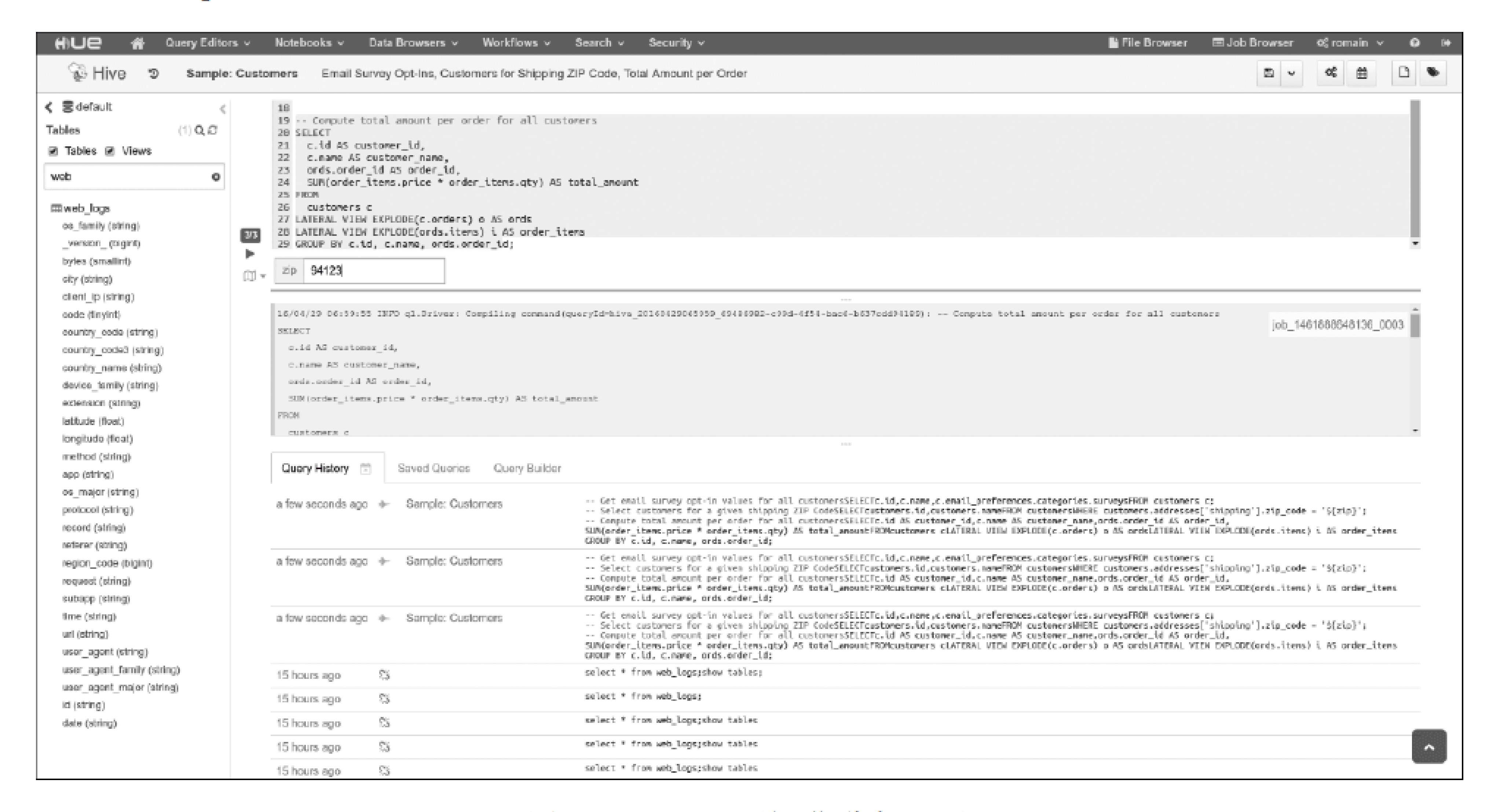


图10-8　Hue可视化分析工具

Hue 提供可视化的数据操作，包括基本的 HDFS 数据上传和下载操作等，回想一下采用 shell 方式进行数据上传、下载的方式(copyFromLocal 等)，这里就不再进行数据上传、下载的操作展示，直接进入数据分析环节。如图 10-9 所示，假定数据已经存放在 HDFS 上，且以 Hive 数据仓库的形式存储，表名是 TDATA。

Showing 146,087,462 Rows out of 146,087,462

passenger_count	trip_distance	pickup_longitude	pickup_latitude	RateCodeID	store_and_fwd_flag	dropoff_longitude	dropoff_latitude	fare_amount	extra	mta_tax	tip_amount	tolls_amount	total_amount
1	0.6	-73.99879455566406	40.739768981933594	1	Y	-73.99201202392578	40.73503112792969	6	0	0.5	0	0	
1	3.9	-74.00989532470703	40.706241607666016	1	N	-73.95157623291016	40.713844299316406	16	0.5	0.5	1.5	0	
3	2.2	-73.95637512207031	40.77815628051758	1	N	-73.96534729003906	40.77195739746094	12	0.5	0.5	3.95	0	1
1	4.1	-73.95417022705078	40.763885498046875	1	N	-73.99440002441406	40.74359893798828	16.5	1	0.5	4.55	0	2
1	2.5	-73.95211791992188	40.77728271484375	1	N	-73.98306274414062	40.76512145996094	12	0	0.5	2.55	0	1
1	5.5	-73.9927978515625	40.76344680786133	1	N	-73.94975280761719	40.82216262817383	16.5	0.5	0.5	3.55	0	2
1	2.3	-73.99193572998047	40.730224609375	1	N	-73.98582458496094	40.756248474121094	17	0	0.5	1.5	0	
1	10.6	-73.8730239868164	40.77397918701172	1	N	-73.98332977294922	40.76285934448242	36	0	0.5	0	5.54	4
2	1.1	-73.99932861328125	40.74406051635742	1	N	-73.98766326904297	40.75961685180664	9.5	0	0.5	2.06	0	1
1	1.4	-73.99172973632812	40.72710418701172	1	N	-73.98027038574219	40.74436950683594	7	0.5	0.5	1.65	0	
1	1.1	-73.97174835205078	40.746124267578125	1	N	-73.96701049804688	40.75979995727539	8	1	0.5	1	0	

Rows 1-11 out of 146,087,462

图 10-9　存放在 HDFS 上的出租车数据样例及其地理分布

(1) 基本数据分析。可用于交通规划部门了解每天的车辆、驾驶员基本信息。

① 总记录条数：

```
SELECT COUNT(*) FROM TDATA;
```

② 总车辆数：

```
SELECT COUNT(medallion) FROM TDATA;
```

③ 总驾驶员数：

```
SELECT COUNT(hack_license) FROM TDATA;
```

(2) 载客数据分析。可用于分析哪些司机或车辆最会接生意。

① 查询当天生意最好的前 20 辆出租车：

```
SELECT medallion, COUNT(*) AS cnt FROM TDATA GROUP BY medallion ORDER BY cnt DESC LIMIT 20;
```

② 查询当天单次载客人数 3 人及以上且接单次数最多的前 20 辆出租车：

```
SELECT medallion, passenger_count, COUNT(*) AS cnt FROM TDATA WHERE passenger_count >=3
GROUP BY medallion, passenger_count ORDER BY cnt DESC LIMIT 20;
```

(3) 载客频率分析。可用于分析哪辆车或司机最繁忙。

① 查询每辆车的载客次数：

```
SELECT medallion, COUNT(medallion) FROM TDATA GROUP BY medallion;
```

② 查询每位司机的载客次数：

```
SELECT hack_license, COUNT(hack_license) FROM TDATA GROUP BY hack_license;
```

(4) 车辆行为分析。可用于分析哪些车或司机更喜欢搭载长途乘客，哪些车每天利用率不高(仅搭载 1 人)。

① 每辆车的平均行驶里程：

```
SELECT medallion, AVG(CAST(trip_distance as double)) FROM TDATA GROUP BY medallion;
```

② 每辆车的搭载一名乘客时平均行驶里程：

```
SELECT medallion, AVG(CAST(trip_distance as double)) FROM TDATA WHERE passenger_count =1
GROUP BY medallion;
```

10.3.3 新闻组数据分析

经典的 20 个新闻组数据集合是收集了大约 20 000 条新闻组文档(http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups)，均匀分布在 20 个不同的集合的一个数据集，是数据挖掘、机器学习领域常用的数据集。本案例旨在基于 MapReduce 计算框架编程实现该新闻组数据的聚类分析。

首先，回顾下需要用到的文本聚类基础知识。词袋模型是在自然语言处理和信息检索中的一种简单假设，最初被用在文本分类中，将文档表示成特征矢量。其基本思想是：假定对于一个文本，忽略其词序和语法、句法，仅仅将其看作是一些词汇的集合，而文本中的每个词汇都是独立的。简单地说就是将每篇文章都看成一个袋子，构成文章的元素都是单词，被装在这个文章“袋子”里，因此称为词袋(bag of words)。

例如有如下两个文档，基于这两个文档构造一个词典。这个词典一共包含 N 个不同的单词，利用词典的索引号，上面两个文档的每一个都可以用一个 N 维向量表示(用整数数字 $0 \sim n$ 表示某个单词在文档中出现的次数)，如图 10-10 所示。

(1) John likes to watch movies. Mary likes movies too.

(2) John also likes to watch football games.

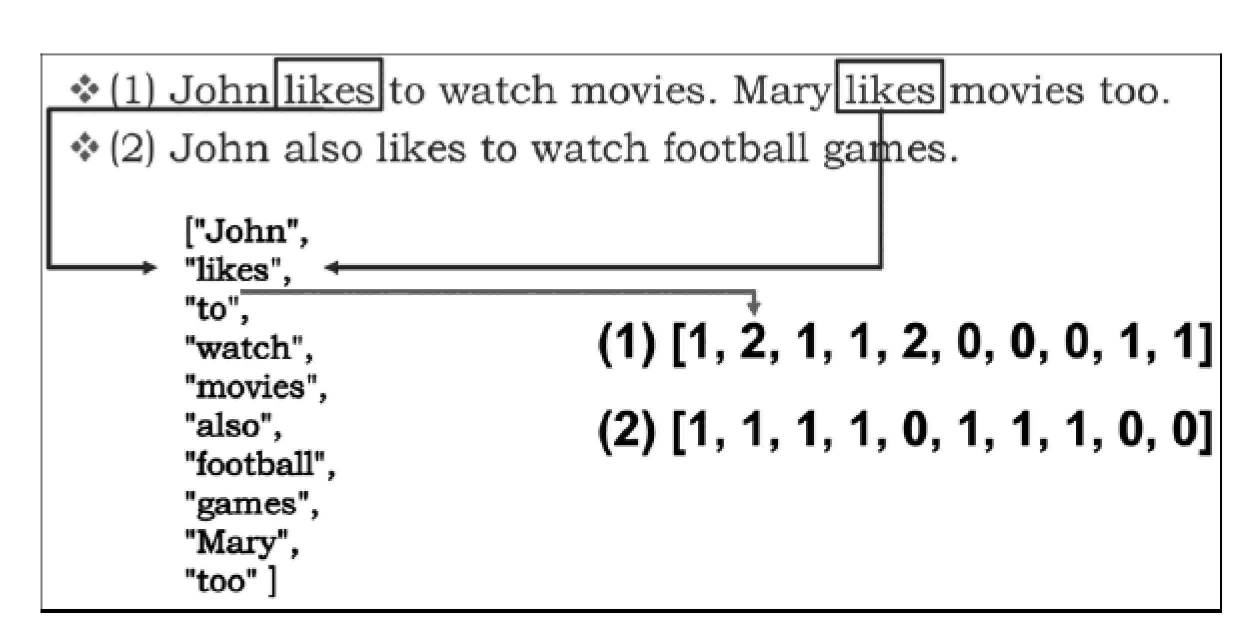


图 10-10 词袋模型生成矢量

为文档生成对应的词袋模型，每一行表示一个文档：

```
dataset= [['John','likes','to','watch','movies','Mary','likes','movies','too'],
          ['John','also','likes','to','watch','football','games']]
```

(1) 生成词汇表：

```
vocabSet = set()
for doc in dataset:
    vocabSet |= set(doc)
```

```
vocabList = list(vocabSet)
```

(2) 为每一个文档创建词袋向量：

```
# 词袋模型
BOW= []
for doc in dataset:
    vec = [0] * len(vocabList)
    for word in doc:
        vec[vocabList.index[word]] +=1
BOW.append(vec)
```

基于 10.2.1 节讲解的 MR *k*-means 算法思想，结合上述词袋模型可实现对新闻组数据的聚类分析。目前，美国 RICE 已开放了相关工具和源代码(http://cmj4.web.rice.edu/MapRedKMeans.html)。在该示例中已按照词袋模型完成了新闻组数据分析，形成了 vectors 向量文件和初始聚类中心文件 clusters，可以直接进行聚类实验，操作步骤如下：

(1) 将 vectors/clusters 上传至 HDFS：

```
hdfs dfs -copyFromLocal vectors /data
hdfs dfs -copyFromLocal clusters /clusters
```

(2) 运行聚类：

```
hadoop jar MapRedKMeans.jar KMeans /data /clusters 3
```

/data 目录存放待聚类数据，/clusters 目录存放初始聚类，3 表示运行三次迭代。

以上操作的输出结果如图 10-11 所示。

图中：

① 处表示在 HDFS 上创建 data 数据目录。

② 处表示在 HDFS 上创建聚类结果目录。

③ 处表示从本地文件系统向 HDFS 上传 vectors 矢量数据。

④ 处表示从本地文件系统向 HDFS 上传初始聚类数据。

⑤ 处表示使用 hadoop 命令调用编译打包的 MRKMeans 聚类程序。

(3) 查看聚类结果：

```
hdfs dfs -copyToLocal /clustersNewXXX/part-r-00000.
```

使用压缩包中的工具查看聚类结果分布：

```
java -jar GetDistribution.jar
Enter the file with the data vectors: vectors
Enter the name of the file where the clusers are loated: part-r-00000
```

在美国莱斯大学提供的工具中的源代码里省略了 *k*-means 执行过程中的 Mapper 和 Reducer 代码，本书将其补充出来并增加了注释部分。完成的示例如下。

在主类入口里面，本示例代码没有使用 10.3.1 节统计分析网页搜索数据时的显式指

```
iTerm Shell Edit View Profiles Toolbelt Window Help                                    22:37
zhangxu@master:~/Downlo...   zhangxu@slave:~ (ssh)
[zhangxu@master KMeans]$ ls
20_newsgroups  clusters  GetCentroids.jar  GetDistribution.jar  KMeans.jar  ProcessCorpus.jar  SeqKMeans  vectors
[zhangxu@master KMeans]$ vi clusters
[zhangxu@master KMeans]$ vi vectors
[zhangxu@master KMeans]$ hdfs dfs -mkdir /data
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zhangxu/hadoop-2.5.2/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zhangxu/hbase-0.98.9-hadoop2/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16/03/06 06:35:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[zhangxu@master KMeans]$ hdfs dfs -mkdir /clusters
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zhangxu/hadoop-2.5.2/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zhangxu/hbase-0.98.9-hadoop2/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16/03/06 06:35:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[zhangxu@master KMeans]$ hdfs dfs -copyFromLocal vectors /data
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zhangxu/hadoop-2.5.2/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zhangxu/hbase-0.98.9-hadoop2/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16/03/06 06:36:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[zhangxu@master KMeans]$ hdfs dfs -copyFromLocal clusters /clusters
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zhangxu/hadoop-2.5.2/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zhangxu/hbase-0.98.9-hadoop2/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16/03/06 06:36:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[zhangxu@master KMeans]$ cd ..
[zhangxu@master Downloads]$ ls
data  hadoop-2.5.2.tar.gz  KMeans  MapRedKMeans-2  mysql-connector-java-5.1.27  mysql-connector-java-5.1.27.tar.gz
[zhangxu@master Downloads]$ cd MapRedKMeans-2/
[zhangxu@master MapRedKMeans-2]$ ls
Hadoop.jar          IIndexedData.java  ISparseArray.java  KMeansMapper.java   LinearIndexedIterator.java  MapRedKMeans.jar         VectorizedObject.java
IDoubleVector.java  IndexedData.java   KMeans.java        KMeansReducer.java  LinearSparseArray.java      SparseDoubleVector.java
[zhangxu@master MapRedKMeans-2]$ hadoop jar MapRedKMeans.jar KMeans /data /clusters 3
```

图 10-11　MR k-means 算法聚类新闻组数据示例

定主类入口的方式——job.setJarByClass(WebSearchMain.class)，而是采用了参数录入的方式，需要在运行时指定数据输入路径。

```
/*
 * import 部分省略
 */
public class MRKMeans {
  static void printUsage() {
      System.out.println ("MRKMeans <input><clusterFileDirectory><numIters>");
      System.exit(-1);
  }
  public static void main (String [] args) throws Exception {
      //args 异常退出
      if (args.length != 3) {
        printUsage ();
        return 1;
      }
      for (int i=0; i <Integer.parseInt (args[2]); i++) {
          //获取配置信息
          Configuration conf=new Configuration ();
          //目录名称
          String dirName;
          //初次交互使用目录名称
          if (i ==0)
            dirName=args[1];
          //使用当前交互名称
```

```
else
  dirName=args[1] +i;
//读取目录下的文件列表
FileSystem fs=FileSystem.get (conf);
Path path=new Path (dirName);
FileStatus fstatus[]=fs.listStatus (path);
int count=0;
for (FileStatus f: fstatus) {
    //忽略"_"开头的文件(Hadoop output)
    if (f.getPath().toUri().getPath().contains ("/_"))
    continue;
    count++;
    conf.set ("clusterInput", f.getPath().toUri().getPath());
}
//确保只有一个聚类文件
if (count !=1) {
    thrownew RuntimeException ("Found more than a single file in the
            clusters directory!");
}
//初始化新的 Job
Job job=new Job(conf);
job.setJobName ("MRKMeans clustering");
//设定输入输出均为 Text
job.setMapOutputKeyClass (Text.class);
job.setMapOutputValueClass (Text.class);
job.setOutputKeyClass (Text.class);
job.setOutputValueClass (Text.class);
//告诉 Hadoop 使用哪个 Mapper 和 Reducer
job.setMapperClass (MRKMeansMapper.class);
job.setReducerClass (MRKMeansReducer.class);
//设置输入输出格式,如何读/写 HDFS
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
//设置输入输出文件
TextInputFormat.setInputPaths (job, args[0]);
TextOutputFormat.setOutputPath (job,new Path (args[1] + (i +1)));
//强制输入分割
TextInputFormat.setMinInputSplitSize (job, 4 * 1024 * 1024);
TextInputFormat.setMaxInputSplitSize (job, 4 * 1024 * 1024);
//设置 jar
job.setJarByClass (KMeans.class);
//设置 Reduce 运行一次
job.setNumReduceTasks (1);
//提交 Job
```

```
            System.out.println ("Starting iteration " +i);
            int exitCode=job.waitForCompletion(true) ? 0 : 1;
            if (exitCode !=0) {
                System.out.println("Job Failed!!!");
                return exitCode;
            }
        }
        return 0;
    }
}
```

MRKMeansMapper 类继承自 Mapper 类，首先验证能否顺利读取带聚类的文件，并设置初始聚类信息。

```
/*
 * import 部分省略
 */
public class MRKMeansMapper extends Mapper<LongWritable, Text, Text, Text>{
    private ArrayList<VectorizedObject>oldClusters=new ArrayList
            <VectorizedObject>(0);
    private ArrayList<VectorizedObject>newClusters=new ArrayList
            <VectorizedObject>(0);
    protected void setup(Context context) throws IOException, Interrupted-
              Exception {
        //打开文件
        Configuration conf=context.getConfiguration();
        FileSystem dfs=FileSystem.get(conf);
        //打开文件失败异常
        if (conf.get("clusterInput") ==null)
            thrownew RuntimeException("no cluster file!");
        //创建 BufferedReader 读取文件
        Path src=new Path(conf.get("clusterInput"));
        FSDataInputStream fs=dfs.open(src);
        BufferedReader myReader=new BufferedReader(new InputStreamReader(fs));
        //读入文件
        String cur=myReader.readLine();
        while (cur !=null) {
            VectorizedObject temp=new VectorizedObject(cur);
            oldClusters.add(temp);
            VectorizedObject newCluster=temp.copy();
            newCluster.setValue("0");
            newClusters.add(newCluster);
            cur=myReader.readLine();
        }
    }
```

```
    public void map(LongWritable key, Text value, Context context) throws
            IOException, InterruptedException {
        //比较矢量数据与当前聚簇中心的距离,划入最近的聚簇内
        String str=paramText.toString();
        VectorizedObject localVectorizedObject=newVectorizedObject(str);
        double d=9.0E99D;
        int i=-1;
        for (int j=0; j <this.oldClusters.size(); j++) {
            //距离判断
            if (localVectorizedObject.getLocation().distance(((VectorizedObject)
               this.oldClusters.get(j)).getLocation()) <d) {
                 i=j;
                 d=localVectorizedObject.getLocation().distance(((VectorizedObject)
                   this.oldClusters.get(j)).getLocation());
            }
        }
        localVectorizedObject.getLocation().addMyselfToHim
            (((VectorizedObject) this.newClusters.get(i)).getLocation());
        ((VectorizedObject) this.newClusters.get(i)).incrementValueAsInt();
    }
    protectedvoid cleanup(Context context) throws IOException,
        InterruptedException {
        …//代码省略
    }
}
```

MRKMeansReducer 类继承自 Reducer 类,对中间聚类结果进行 Reduce 操作,在满足迭代终止条件时停止并输出最终聚簇信息。

```
/*
 * import 部分省略
 */
public class MRKMeansReducer extends Reducer<Text, Text, Text, Text>{
    //put any private data structured you need here!
    publicvoid reduce(Text key, Iterable<Text>values, Context context)
        throws IOException, InterruptedException {
        VectorizedObject localVectorizedObject1=null;
          for (Text localText : paramIterable) {
            if (localVectorizedObject1 ==null) {
              localVectorizedObject1=newVectorizedObject(localText.toString());
            } else {
            VectorizedObject localVectorizedObject2=new
                VectorizedObject(localText.toString());
            localVectorizedObject2.getLocation().addMyselfToHim
```

```
                (localVectorizedObject1.getLocation());
            localVectorizedObject1.addToValueAsInt
                (localVectorizedObject2.getValueAsInt());
            }
        }
        this.newClusters.add(localVectorizedObject1);
    }
    protected void cleanup(Context context) throws IOException,
        InterruptedException {
        …//代码省略
    }
}
```

10.4 小 结

本章介绍了大数据并行算法的基本思想和方法，以离线处理和交互式处理技术的代表 MapReduce 和 Spark 为例进行了分析。最后通过网页搜索日志分析、出租车数据统计分析和新闻组数据分析 3 个真实案例对大数据分析和程序设计进行了详细讲解。通过本章的学习，希望为读者在大数据并行算法设计与编程开发奠定基础。

- 数值计算是指基于代数关系的运算，包括矩阵运算、多项式求解、解线性方程组等计算问题，属于数值分析的范畴。
- 非数值计算是指基于比较关系的运算，包括排序、选择、搜索、匹配以及图论等方面的计算问题，属于符号处理的范畴。
- 同步算法是指某些进程必须等待别的进程的并行算法。
- 异步算法是指各个进程的执行一般不必互相等待的并行算法，进程的通信通过动态读取(修改)共享存储器的全局变量实现。
- 分布算法是指由通信链路连接的多个场点或节点协同完成问题求解的并行算法。

10.5 习 题

1. 简述如何基于 MapReduce 设计并行算法。
2. 简述如何基于 Spark 设计并行算法。
3. 简述传统 k-means 算法的局限性并分析其时间、空间复杂度。
4. 简述 MapReduce k-means 算法的基本思想。
5. 结合你的理解，谈谈 MapReduce 和 Spark 开发并行算法的区别。
6. 简述 MapReduce 并行算法设计主类应包含的常见内容。
7. 简述词袋模型的基本思想。
8. 结合实际，试分析词袋模型在文本分析之外的可能应用领域。
9. 简述什么是并行计算，并从不同的角度分析其分类。

10.6 参考文献

[1] 国务院. 国务院关于促进云计算创新发展培育信息产业新业态的意见：国发[2015]5 号. 2015.

[2] 罗乐，刘轶，钱德沛. 内存计算技术研究综述[J]. 软件学报，27(8)，2016：2147-2167.

[3] Zaharia M，Chowdhury M，Franklin M J，et al. Spark：Cluster Computing with Working Sets. In：Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing (HotCloud 2010). Berkeley：USENIX Association，2010，15(1)：10.

[4] Valant LG. A Bridging Model for Parallel Computation. Communications of the ACM，1990，33(8)：103-111. [doi：10.1145/79173.79181]

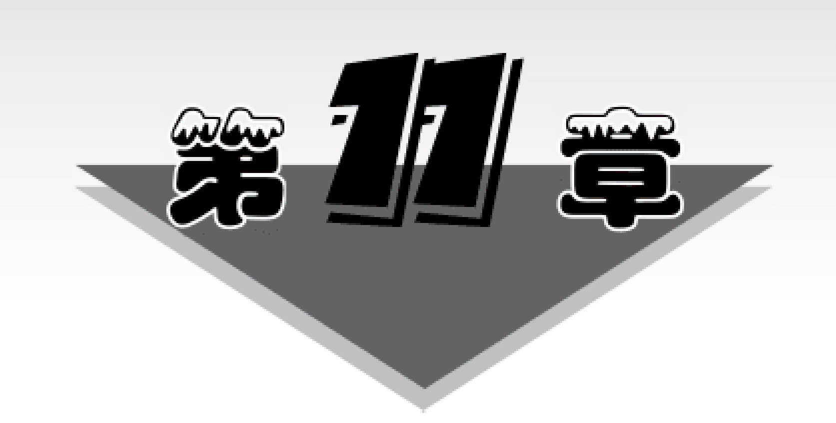
[5] Malewicz G，Austern M H，Bik A J C，et al. Pregel：A System for Large-Scale Graph Processing. In：Proc. of the 2010 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press，2010：135-146. [doi：10.1145/1807167.1807184]

[6] Gonzalez J E，Low Y，Gu H，et al. PowerGraph：Distributed Graph-Parallel Computation on Natural Graphs. In：Proc. of the 10th USENIX Conf. on Operating Systems Design and Implementation. Hollywood，2012：17-30.

[7] Shao B，Wang H，Li Y. Trinity：A Distributed Graph Engine on a Memory Cloud. In：Proc. of the 2013 Int'l Conf. on Management of Data. ACM Press，2013：505-516. [doi：10.1145/2463676.2467799]

[8] 潘巍，李战怀. 大数据环境下并行计算模型的研究进展[J]. 华东师范大学学报，2014，(5)：43-54.

[9] 陈国良. 并行算法的设计与分析. 2 版. 北京：高等教育出版社，2002.



大数据挖掘及应用展望

当"云计算""物联网"这样的概念对大众而言还是一知半解的时候,"大数据"横空出世,发展十分迅猛。例如,为了减少火车脱轨造成的伤亡,火车上安装了各种传感器来收集各个部位的运行情况数据,以此来检测存在安全隐患的器件。当然,这还远远没有达到智能的水平,还需要对铁轨乃至整个交通系统都能够进行实时的数据采集,甚至包括天气状况。当把这些信息加入到火车运行过程的所有数据里面,一个大数据的智能挖掘问题就产生了。如今,我们身处数据海洋,几乎所有事物都时时刻刻地产生着数据,环境、金融、电商、医疗、电信、办公、社交媒体……大量的实时数据影响着我们的生活、工作乃至社会的发展、人类的进步,因此大数据挖掘及应用引起了政府、科技界和企业界的高度重视。随着相关应用的不断涌现,数据挖掘领域的新知识及新理论也在不断地出现。

本章主要关注大数据挖掘及应用的趋势和研究前沿。从大数据时代的发展回顾与展望(11.1 节)开始,随后介绍大数据挖掘的新数据类型和大数据挖掘的新方法,最后对大数据的研究和应用发展进行展望。

11.1 大数据时代的发展回顾与展望

本节回顾大数据发展的历史,并对未来的智能分析进行展望。首先回顾大数据发展的重大事件,然后简要介绍大数据分析从小数据到大数据的发展历程,最后结合人工智能的发展简要分析大数据挖掘的广泛应用和前景。

11.1.1 大数据发展回顾

自 1936 年图灵关于可计算的论文发表以来,已经经历了计算机的诞生、软件工程的诞生以及后来互联网、万维网和网络科学、维基百科的出现。近年来,云计算、大数据的发展,使人类社会真正进入了信息社会时代、网络社会时代和大数据时代。科学技术及互联网的发展推动着大数据时代的来临。各行各业每天都在产生数量巨大的数据碎片,数据计量单位已从 B、KB、MB、GB、TB 发展到 PB、EB、ZB、YB 甚至 BB、NB、DB 来衡量。大数据时代,数据的采集也不再是技术问题。更重要的是面对如此庞大的数据,我们怎样才能发现其内在规律。大数据必然无法用人脑来推算、估测,或者用单台计算机进行处理,必须依托云计算的分布式处理、分布式数据库、云存储和虚拟化技术。

Hadoop 自诞生至今已有 10 年,它改变了企业对数据的存储、处理和分析的过程,加速

了大数据的发展,形成了极其火爆的技术生态圈,并得到十分广泛的应用。在过去的十年,世界各国政府、学术界、产业界对大数据投入了极大的关注并产生了一系列理论和时间成果:

- 2005 年,Hadoop 项目诞生,它由两项关键服务构成:采用 Hadoop 分布式文件系统(HDFS)的可靠数据存储服务,利用 MapReduce 的高性能并行数据处理服务。
- 2008 年,《自然》杂志出版大数据专辑,掀起了学术界对大数据科学技术研究的热潮。
- 2008 年,美国发布白皮书《大数据计算:在商务、科学和社会领域创建革命性突破》,提出:大数据真正重要的是新用途和新见解,而非数据本身。
- 2009 年,联合国全球脉动项目利用手机和社交网站的数据源来分析预测从螺旋价格到疾病暴发的问题;美国通过启动 Data.gov 网站开放政府数据大门,向公众提供各种各样的政府数据。
- 2011 年,IBM 公司的沃森每秒可扫描并分析 4TB(约 2 亿页文字量)的数据量,在美国著名智力竞赛电视节目"危险边缘"(Jeopardy)上击败两名人类选手而夺冠,首次实现了利用大数据对人类的超越。
- 2011 年,全球知名咨询公司麦肯锡(McKinsey&Company)肯锡全球研究院(MGI)发布报告《大数据:创新、竞争和生产力的下一个新领域》,指出:大数据已经渗透到当今每一个行业和业务领域,成为重要的生产因素。
- 2011 年,《科学》出版专刊 *Dealing with Data*。
- 2011 年,中国工业和信息化部发布的物联网"十二五"规划上,信息处理技术作为 4 项关键技术创新工程之一被提出来,其中包括海量数据存储、数据挖掘、图像视频智能分析,这些都是大数据的重要组成部分。
- 2012 年,在瑞士达沃斯召开的世界经济论坛上,大数据是主题之一,会上发布的报告《大数据,大影响》(*Big Data, Big Impact*)宣称,数据已经成为一种新的经济资产类别,就像货币或黄金一样。
- 2012 年,美国奥巴马政府发布《大数据研究和发展倡议》,宣布将 2 亿美元投资于大数据领域,是大数据技术从商业行为上升到国家科技战略的分水岭,政府将数据定义为"未来的新石油"。
- 2012 年,联合国发布了一份关于大数据政务的白皮书,总结了各国政府如何利用大数据更好地服务和保护人民。指出:在一个数据生态系统中,个人、公共部门和私人部门各自的角色、动机和需求;政府如果能合理分析所掌握的数据资源,将能"与数俱进",快速应变。
- 2014 年,美国白宫发布了全球"大数据"白皮书的研究报告《大数据:抓住机遇、守护价值》。鼓励使用数据推动社会进步。
- 2014 年,Spark 1.0.0 版发布。Spark 拥有 Hadoop MapReduce 所具有的优点,Job 中间输出结果可以保存在内存,启用了内存分布数据集。
- 2015 年,中国国务院发布《关于印发促进大数据发展行动纲要的通知》(国发[2015]50 号)。
- 2016 年,《大数据标准化白皮书(2016 版)》发布。
- 2016 年,国务院印发《"十三五"国家战略性新兴产业发展规划》,云计算、大数据、人工智能、"互联网+"等热词在规划中多次出现。

- 2017 年，工业和信息化部正式发布了《大数据产业发展规划（2016—2020 年）》，以强化大数据产业创新发展能力为核心，明确了强化大数据技术产品研发、深化工业大数据创新应用、促进行业大数据应用发展、加快大数据产业主体培育、推进大数据标准体系建设、完善大数据产业支撑体系、提升大数据安全保障能力等 7 项任务，提出大数据关键技术及产品研发与产业化工程、大数据服务能力提升工程等 8 项重点工程，研究制定了推进体制机制创新、健全相关政策法规制度、加大政策扶持力度、建设多层次人才队伍、推动国际化发展等 5 项保障措施。
- 2017 年 7 月，国务院发布《新一代人工智能发展规划》，同年 11 月，科技部召开新一代人工智能发展规划暨重大科技项目启动会，宣布成立新一代人工智能发展规划推进办公室。同时，会议宣布分别依托百度、阿里、腾讯、科大讯飞建设自动驾驶、城市大脑、医疗影像、智能语音领域的国家人工智能开放创新平台。

经过 10 年的积累和沉淀，大数据在理论、技术方面都取得了极大的飞跃，并逐渐在各个领域崭露头角，发挥作用。世界上有各种不同来源的大数据，如自然大数据、生命大数据、社交大数据等。如图 11-1 所示，以医学大数据为例，让我们看看现代医学是如何治疗疾病的。医生的诊断要依赖于各种检查和化验的数据结果，表现为文字描述、图表展示、静态医学影像、动态医学影像甚至高维可视化数据等。可以说检查就是诊治，没有科学的大数据检测手段，就不可能有高质量的医疗服务。

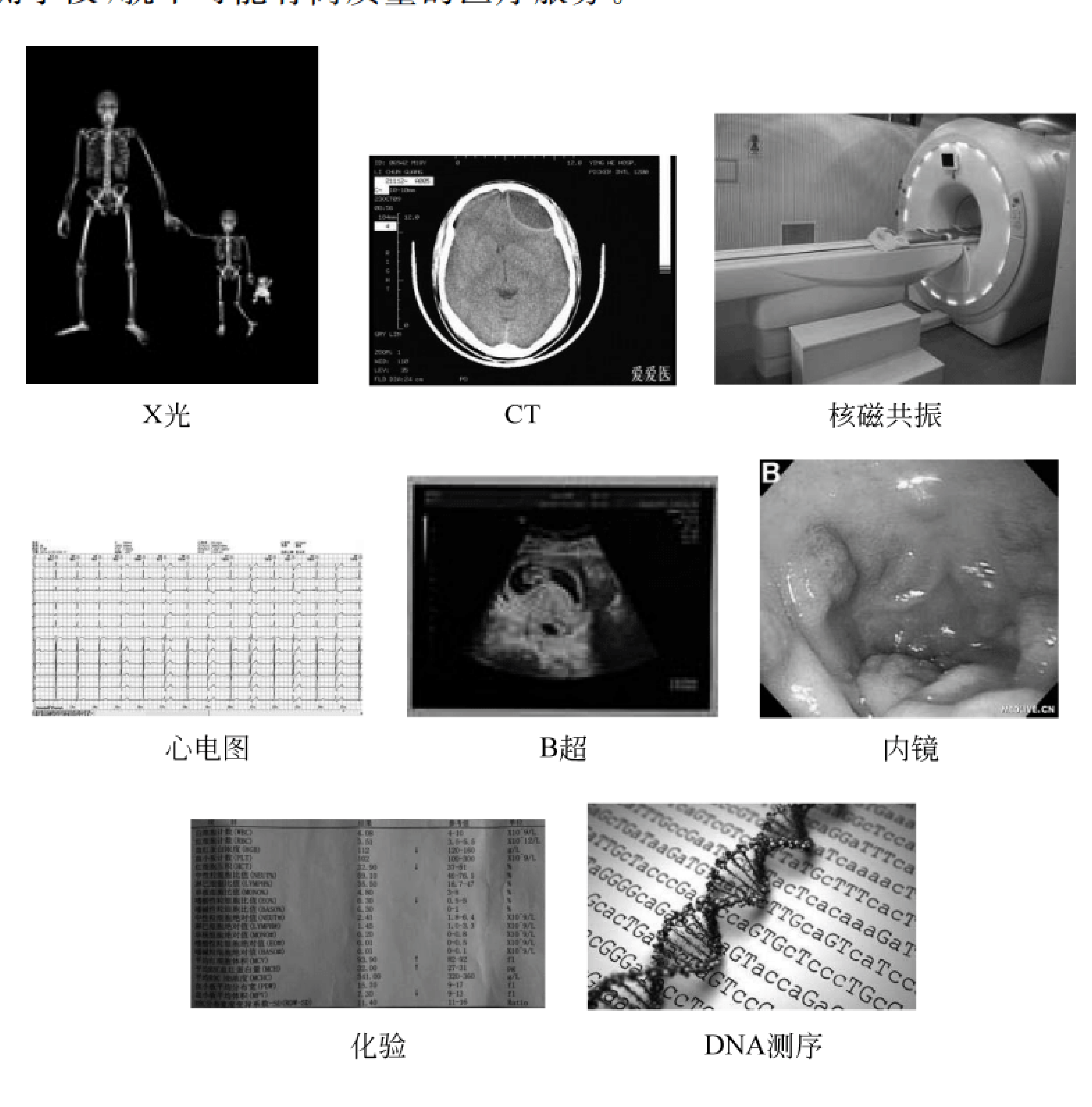


图 11-1　医学监测技术

再来看看近年来的医学诺贝尔奖都发给了谁？多数获奖者是器械发明人以及能够从检测数据中发现价值的人(图 11-2)。技术设备的发展加快了医学进步的步伐。



威廉·埃因托芬：心电图发明人

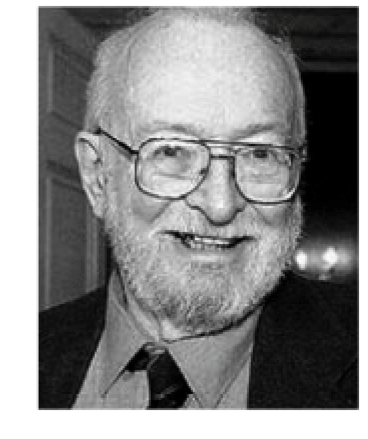


赫尔曼·约瑟夫·马勒：X射线辐射治疗发明人





阿兰·麦克莱德·科马克，高弗雷·豪斯费尔德：计算机辅助X射线断层摄影(CT)发明人





保罗·劳特布尔，彼得·曼斯菲尔德：核磁共振成像发明人

图 11-2　著名医学检测技术科学家

11.1.2　从"小"到"大"的数据分析处理

数据分析处理经历了数据分析、数据挖掘、海量数据挖掘、大数据挖掘等几个阶段，经历了从小数据到大数据的发展历程。数据分析的研究已经有几百年的历史，可以追溯到 1763 年 Thomas Bayes 提出的 Bayes 理论，它是数据挖掘和概率论的基础。回归分析也是数据分析的重要数学理论工具之一。20 世纪，计算机时代的到来让海量数据的收集与处理成为了可能。最近几十年发展起来的神经网络、进化计算、遗传算法和支持向量机等理论模型都是数据分析处理的有效模型。在各个领域中取得了很多的成功：

- 1763 年，Thomas Bayes 的论文在他去世后发表，他提出的 Bayes 理论将当前概率与先验概率联系起来，帮助理解基于概率估计的复杂现况，成为数据挖掘和概率论的基础。
- 1805 年，Adrien-Marie Legendre 和 Carl Friedrich Gauss 使用回归分析(最小二乘法)确定了天体(彗星和行星)绕行太阳的轨道。回归分析的目标是估计变量之间的关系，成为数据挖掘的重要工具之一。
- 1936 年，Alan Turing 发表论文 *On Computable Numbers on Computable Numbers, with an Application to the Etscheidungs Problem*，提出了通用机，计算机时代即将到来，它让海量数据的收集和处理成为可能。
- 1943 年，Warren McCullon 和 Walter Pitts 发表论文 *A Logical Calculus of the Ideas Immanent in Nervous Activity*，提出神经网络的概念模型，阐述了网络中神经元的概念，每一个神经元做三件事：接收输入、处理输入和生成输出。

- 1965年，Lawrence J. Fogel成立了Decision Science公司，这是第一家专门将进化计算应用于解决现实世界问题的公司。
- 1975年，John Henry Holland出版*Adaptation in Natural and Artificial Systems*，成为遗传算法领域具有开创意义的著作。
- 1992年，Berhard E. Boser，Isabelle M. Guyon和Vladimir N. Vapnik提出改进的支持向量机，充分考虑到非线性分类问题。

20世纪70年代，随着数据库管理系统的成熟，存储和查询巨量数据(高达千万亿字节)成为可能。数据仓库的出现，促使用户的思维方式从面向事务处理转变到数据分析处理。在1989年召开的国际人工智能联合学术会议上提出"数据库中的知识发现"(KDD)这一学术术语，由此诞生了数据挖掘这一新的研究领域，ACM、IEEE等学术机构纷纷组织学术会议，推动数据挖掘的研究发展，掀起了数据挖掘的研究热潮。2015年，美国白宫任命了第一位首席数据科学家和制定数据策略的副首席技术官，政府管理越来越依赖于大数据的科学分析。

11.1.3 大数据的智能分析与挖掘

自从1956年达特茅斯会议提出"人工智能"这一术语以来，人工智能这一学科经历了60年的发展，形成了符号主义、连接主义、行为主义三大学派，分别从不同角度研究和探讨如何用机器模拟智能的问题。人们不禁要问，机器真的能够实现智能吗？为了测试机器是不是具备人类智能的能力，图灵1950年设计了图灵测试：如果计算机能够在5分钟内回答人类测试者提出的一系列问题，且超过30%的回答让测试者误认为是人类在回答，则计算机通过测试。现在计算机不但通过了图灵测试，还取得了一系列超乎想象的成功。在一系列智能挑战赛中，机器战胜了人类，而且机器智能在多个社会领域中取得了成功应用(图11-3)。

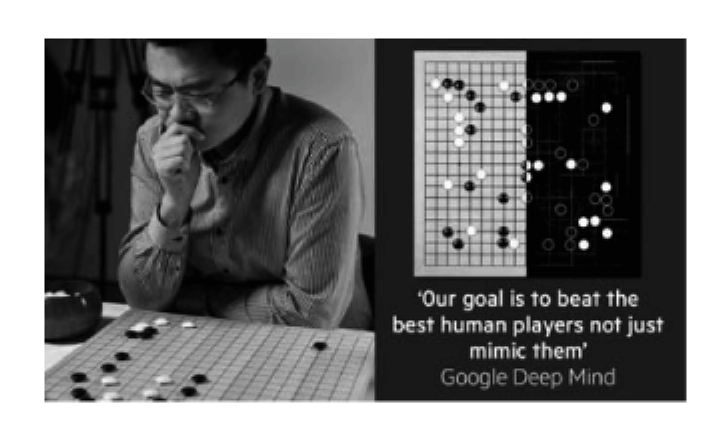
- 1997年5月11日是机器与人类之间挑战赛的历史性一天。IBM公司的深蓝超级计算机在正常时限的比赛中首次击败了当时世界排名第一的国际象棋大师卡斯帕罗夫，标志着国际象棋进入了新的时代。
- 2011年，由IBM公司和美国得克萨斯大学联合研制的超级计算机沃森在美国最受欢迎的智力竞赛电视节目《危险边缘》击败该节目历史上最成功的选手Ken和Brad成为新的王者。
- 2015年，在德国举办的汉诺威消费电子信息及通信博览会的开幕式上，马云通过扫脸取代了传统的密码支付认证，为德国总理默克尔从淘宝网上购买了1948年的汉诺威纪念邮票，开启了刷脸支付时代的大门。
- 2015年，讯飞听见实现了全球首次会议字音同步直播。
- 2016年1月，在没有任何让子的情况下，AlphaGo以5∶0完胜欧洲围棋冠军、职业围棋二段选手樊麾，在围棋人工智能领域实现了一次史无前例的突破。紧接着，AlphaGo又在全世界的高度关注下4∶1战胜了世界排名第一的韩国围棋选手李世石。



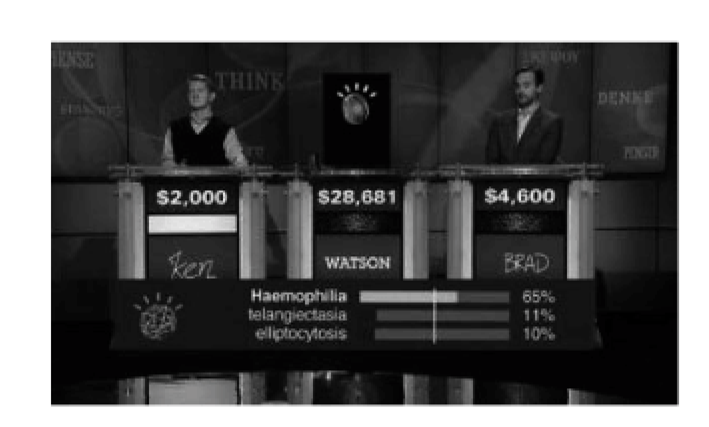
深蓝战胜国际象棋大师
卡斯帕罗夫, 1997



马云刷脸支付, 2015



谷歌AlphaGo击败欧洲围棋
冠军樊麾, 2016



沃森在美国Jeopardy (危险边缘)
节目中击败人类选手, 2011



讯飞听见字音同步直播, 2015



美交管局认定谷歌自动驾驶
系统为“驾驶员”, 2016

图 11-3　人工智能技术的典型应用事件

- 2016 年 2 月，美国国家公路安全管理局认定 Google 自动驾驶汽车内部的计算机可视为“驾驶员”。
- 2016 年底，神秘围棋大师 Master 横空出世横扫人类围棋高手，豪取 60 连胜。
- 2017 年 1 月 6 日第四季《最强大脑》节目中，“百度大脑”现场挑战名人堂选手的“最强大脑”，在图像和语音识别等领域一决高下。最终百度大脑的人工智能机器人“小度”以 3∶2 获胜，上演了中国版“人机大战”。

这些成功又掀起新一轮的人工智能热潮，人工智能正迎来最受关注的发展时代，Master 在围棋比赛中能找到新的规律，甚至打破人类既有的规律体系。从这个意义上来讲，人工智能就是时代的“望远镜”，能够帮助我们改善决策的方式，以前所未有的方法解决问题。近年来不断发生的人机大战让人们深切感受到了人工智能给人类智慧带来的挑战与压力，有助于人类自身认清智能的边界，谨慎理解人工智所蕴含的机遇与风险。

那么，机器智能最终会不会超越人类呢？对此，社会上有许多讨论。归纳起来，大致有三种观点。第一种观点是超越派，认为机器智能最终将超越人类。第二种观点是无限

趋近派，认为机器智能会永远接近人类智能，但不会超越。第三种观点是中立和已经发生派，认为机器智能与人类智能充分融合。不管哪种观点，机器智能作为人类智能的高级技术工具之一，已经取得了人们的认可。科学技术的发展直观体现在人类使用工具手段的进步，机器战胜人，实际上是一个人或者说一群人利用机器和科技手段战胜了另外一个或者一群人。因此我们没有必要杞人忧天，担心有一天机器智能会战胜人类智能，甚至机器会消灭人类。当然，避免将人工智能技术和智能机器应用于危害人类的领域，也需要引起社会各界的关注，就像人类需要和平利用核能技术，但同时又要限制核武器一样。

大数据智能不只体现在机器人上，还表现在智能数据分析对决策的支持上。从数据的采集到智能决策的制定，需要经过数据的感知与获取、信息的编码存储与传递、知识的认知与理解、智能判断与决策行动等多个阶段，而且这还可能是一个循环往复的复杂过程，每个阶段都需要通过智能技术来实现，这也是对大数据分析与处理的挑战。如图 11-4 所示，在云计算和大数据时代，我们面对的数据产生设备是多样的、复杂的，甚至人类本身也在产生数据，如何实现不同多源、异构数据的整合与加工，将数据加工成我们可以读懂的信息是第一个阶段。随着信息的累积，人们已难以辨识繁多信息中的有用知识。通过数据挖掘等技术手段从信息中提取有用的知识，并采用可视化等技术手段进行解读，能够为政府、企事业单位提供有力的决策支持。

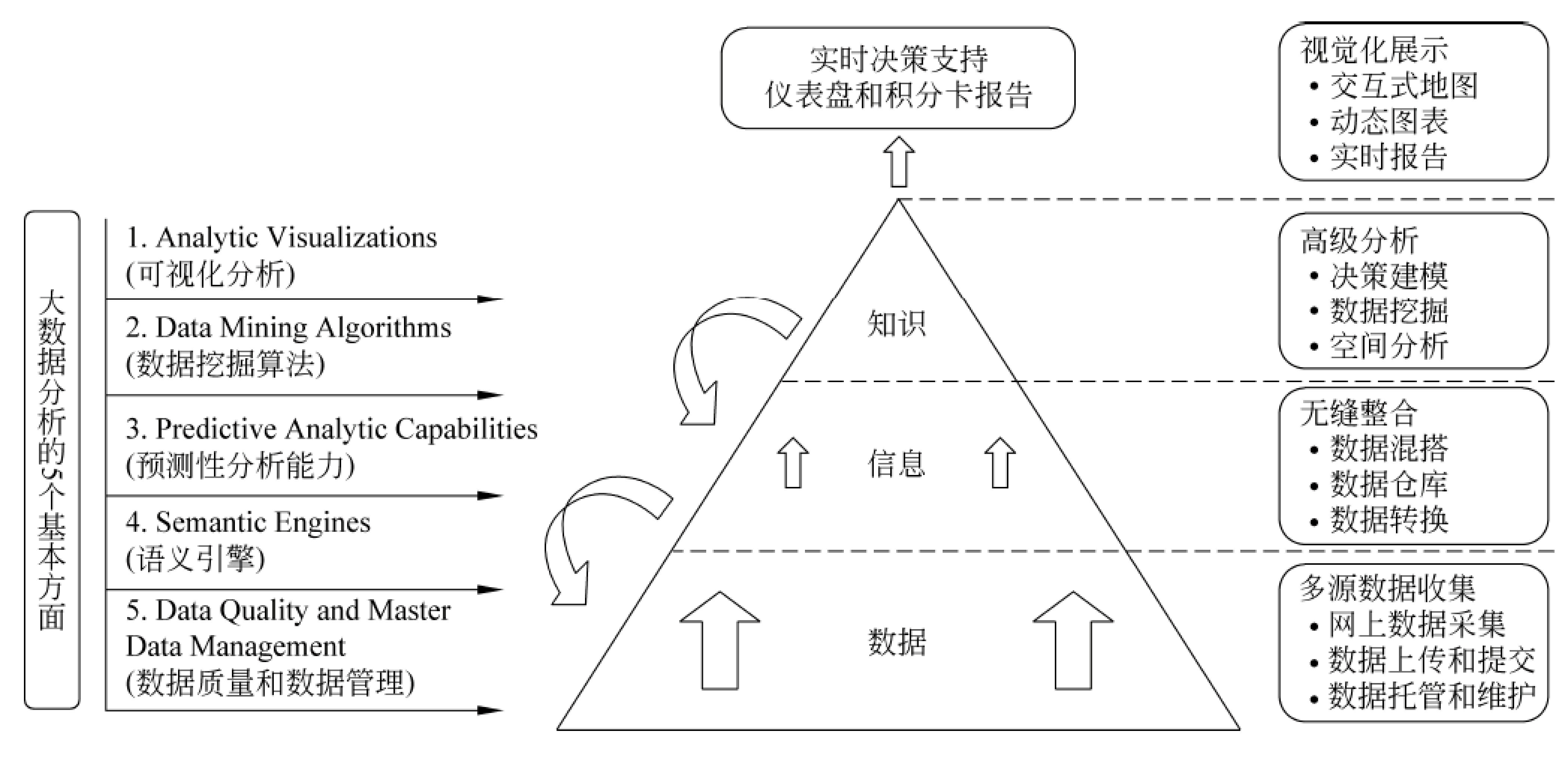


图 11-4　从数据到决策的循环过程

- 传统数据大多被存储在本地，并非全部公开数据资源，例如市场调研数据、企业数据、生产数据、制造数据、消费数据、医疗数据、金融数据等。把握数据资源的企业或行业必然成为大数据的直接受益者。
- 移动互联网的快速发展使搜索引擎及智能手机等移动设备成为重要的数据入口。社交网络、电子商务以及各类应用 APP 等将分散的“小数据”变成“大数据”。
- 物联网的发展能够实现“万物互联”，所有事物产生的信息都是数据，所有事物之

间都具有“数据化”的联系。

通过本书的学习，我们知道大数据的关注点已经不再仅仅是数据量大了，而最重要的是对大数据进行分析和挖掘，只有通过智能分析才能获取深入的、有价值的信息，并将这些信息加工成人们可快速、准确理解的知识。

11.2 大数据中的新数据类型

本节主要介绍非结构化大数据、半结构化大数据和空间大数据等大数据挖掘中会遇到的新数据类型，帮助读者更清楚地认识未来运用大数据技术可分析的内容。

由于能够处理多种数据结构，大数据能够在最大程度上利用互联网上记录的人类行为数据进行分析。大数据出现之前，计算机所能够处理的数据都需要前期进行结构化处理，并记录在相应的数据库中。但大数据技术对于数据的结构要求大大降低，互联网上人们留下的社交信息、地理位置信息、行为习惯信息、偏好信息等各种维度的信息都可以实时处理，立体完整地勾勒出每一个个体的各种特征。在大数据时代，随着信息技术的发展和“互联网＋”的普及，数据以不同的方式在自动产生并被收集，例如：

- 过去，一些记录是以模拟形式存在的，或者以数据形式存在但是存储在本地，不是公开数据资源，没有开放给互联网用户，例如音乐、照片、视频、监控录像等影音资料。现在这些数据不但数据量巨大，并且共享到了互联网上，面对所有互联网用户，其数量之大是前所未有的。第 1 章图 1-1 中提到，Facebook 每分钟有 21 万多张照片上传或被传播，形成了海量的数据，以每张照片 500KB 计算，一小时就会产生 6TB 的数据。
- 移动互联网出现后，移动设备的很多传感器收集了大量的用户点击数据和行为数据。例如，iPhone6 中内置的相机传感器、声波传感器、温度传感器、压力传感器、气压传感器、加速度传感器等，它们每天产生大量的点击数据，形成用户大量行为数据，更不用说用户利用手机访问网络产生的交互数据、日志数据等。
- 在高德、百度、Google 等电子地图和导航应用出现后，产生了大量的地理静态数据和用户动态数据。这些数据不同于传统数据，传统数据代表一个属性或一个度量值，但是这些地图产生的数据代表着一种行为、一种习惯，这些数据经频率分析后会产生巨大的商业价值。基于地图产生的数据流是一种新型的数据类型，在过去是不存在的。
- 进入了社交网络的年代后，互联网行为主要由用户参与创造，大量的互联网用户创造出海量的社交行为数据，这些数据是过去未曾出现的，其揭示了人们行为特点和生活习惯。
- 电子商务的崛起产生了大量网上交易数据，包含支付数据、查询行为、物流运输、购买喜好、点击顺序、评价行为等，其次是信息流和资金流数据。

大数据挖掘中的新数据类型主要有以下 3 类：

(1) 非结构化数据。

非结构化数据库是指其字段长度可变，并且每个字段的记录又可以由可重复或不可

重复的子字段构成的数据库，用它不仅可以处理结构化数据（如数字、符号等信息），而且更适合处理非结构化数据（文本、图像、声音、影视、超媒体等信息）。

非结构化 Web 数据库主要是针对非结构化数据而产生的，与以往流行的关系数据库相比，最大的区别在于它突破了关系数据库结构定义不易改变和数据定长的限制，支持重复字段、子字段以及变长字段，并实现了对变长数据和重复字段进行处理和数据项的变长存储管理，在处理连续信息（包括全文信息）和非结构化信息（包括各种多媒体信息）中有着传统关系型数据库无法比拟的优势。

（2）半结构化数据。

半结构化数据就是介于完全结构化数据（如关系型数据库、面向对象数据库中的数据）和完全无结构的数据（如声音、图像文件等）之间的数据，因其层次性、自述性、动态可变性等特点，被广泛应用在互联网、异构数据集成和交换等领域。HTML 文档、XML 文档、SGML 文档、Web 数据等就属于半结构化数据。这些数据一般是自描述的，数据的结构和内容混在一起，没有明显的区分。

（3）空间大数据。

与空间信息或位置相关的大数据统称为空间大数据，包括以下几类：

- 地理数据。指直接或间接关联着相对于地球的某个地点的数据，包括自然地理数据和经济社会数据，例如地貌数据、土壤数据、水温数据、植被数据等，特点是体量大，较为规则化，变化较慢。
- 轨迹数据。指通过 GNSS 等测量手段以及网络签到等方式获得的用户活动数据，可以被用来反映用户的位置和社会偏好，例如个人轨迹数据、群体轨迹数据、车辆轨迹数据等，特点是数据体量大，信息碎片化，准确性较低，半结构化。
- 空间媒体数据。包含位置的数字化的文字、图形、图像、视频影像等媒体数据，主要来源于移动社交网络、微博等新型互联网应用，例如微信、微博等产生的数据，特点是数据体量大，数据类型多样，非结构化为主。

空间大数据为数据分析、知识发现和决策带来了良好的机遇，同时，其对数据处理能力也提出了重大挑战。空间数据作为整体纳入到统一的空间数据仓库中，能够克服传统地理信息数据存储和数据分析的过程分离，提升空间数据的处理能力。然而，如此庞大规模的空间大数据对现有的数据仓库从存储、处理和分析等各方面都提出了新的挑战。大数据平台下的空间联机分析处理（Spatial OLAP）是一种崭新的决策支持工具，提供 GIS 的地图可视化与非空间数据的交互和分析能力，通过基于地图的上卷、下钻、切片、切块等操作以及空间数据特有的地图覆盖等功能，实现对空间大数据进行可视化的、多视角的、多层次的、多侧面的查询分析；支持空间大数据的多种维度之间、多个数据粒度之间、多个专题时间、多个时间之间的 OLAP 操作，实现在地图和多维图标上同步显示多层级聚集组成的而多维模式，在线、快速实现时空分析和挖掘。

信息基础设施发展各阶段的空间大数据特征对比如表 11-1 所示。

表 11-1 信息基础设施发展各阶段的空间大数据特征对比

比较维度	发展阶段			
	第一代	第二代	第三代	新一代
数据表达模型	文件	文件/关系数据模型	关系模型/空间数据模型	结构化数据模型/非结构化数据模型
数据存储模型	基于主机、个人计算机	基于局域网 C/S 结构	基于 Internet 的 B/S 架构	基于云存储(HDFS)
数据计算模型	以计算为中心	以计算为中心	向以存储为中心转换	分布式并行计算模型(MapReduce)
数据服务模型	系统之间独立	系统之间独立	向共享和协同方向发展	向云服务平台发展
应用开发模型	无	应用编程接口模式	向开发平台方向过渡	向开放云平台过渡
应用服务模型	购买软件	购买软件	购买服务	免费服务(更关注用户隐私)

11.3 大数据挖掘的新方法

本节主要介绍深度学习、知识计算、社会计算和特异群组挖掘等大数据挖掘的新方法,使读者掌握大数据时代的数据挖掘新技术。

11.3.1 深度学习

1956 年,几个计算机科学家相聚在达特茅斯会议(Dartmouth Conferences),提出了“人工智能”的概念。其后,人工智能就一直萦绕于人们的脑海之中,并在科研实验室中慢慢孵化。机器学习作为实现人工智能的一种重要方式,最基础的是运用算法来分析数据,从中学习、测定或预测现实世界某些事。然而,这些早期机器学习方法都没有实现通用人工智能的最终目标,甚至没有实现狭义人工智能的一小部分目标。事实证明,多年来机器学习的最佳应用领域之一是计算机视觉,尽管它仍然需要大量的手工编码来完成工作。

深度学习的概念源于人工神经网络的研究,含多隐层的多层感知器就是一种深度学习结构。深度学习通过组合低层特征形成更加抽象的高层表示属性类别或特征,以发现数据的分布式特征表示。近年来,深度学习在语音、图像以及自然语言理解等应用领域取得一系列重大进展。从 2009 年开始,微软研究院的 Dahl 等人率先在语音处理中使用深度神经网络(DNN),将语音识别的错误率显著降低,从而使得语音处理成为成功应用深度学习的第一个领域。在图像领域,2012 年,Hinton 等人使用深层次的卷积神经网络(CNN)在 Image Net 评测上取得巨大突破,将错误率从 26%降低到 15%,重要的是,这个模型中并没有任何手工构造特征的过程,网络的输入就是图像的原始像素值。在此之后,采用类似的模型,通过使用更多的参数和训练数据,Image Net 评测的结果得到进一步改

善，错误率下降至2013年的11.2%。

人工智能、机器学习、深度学习的发展进程如图11-5所示。

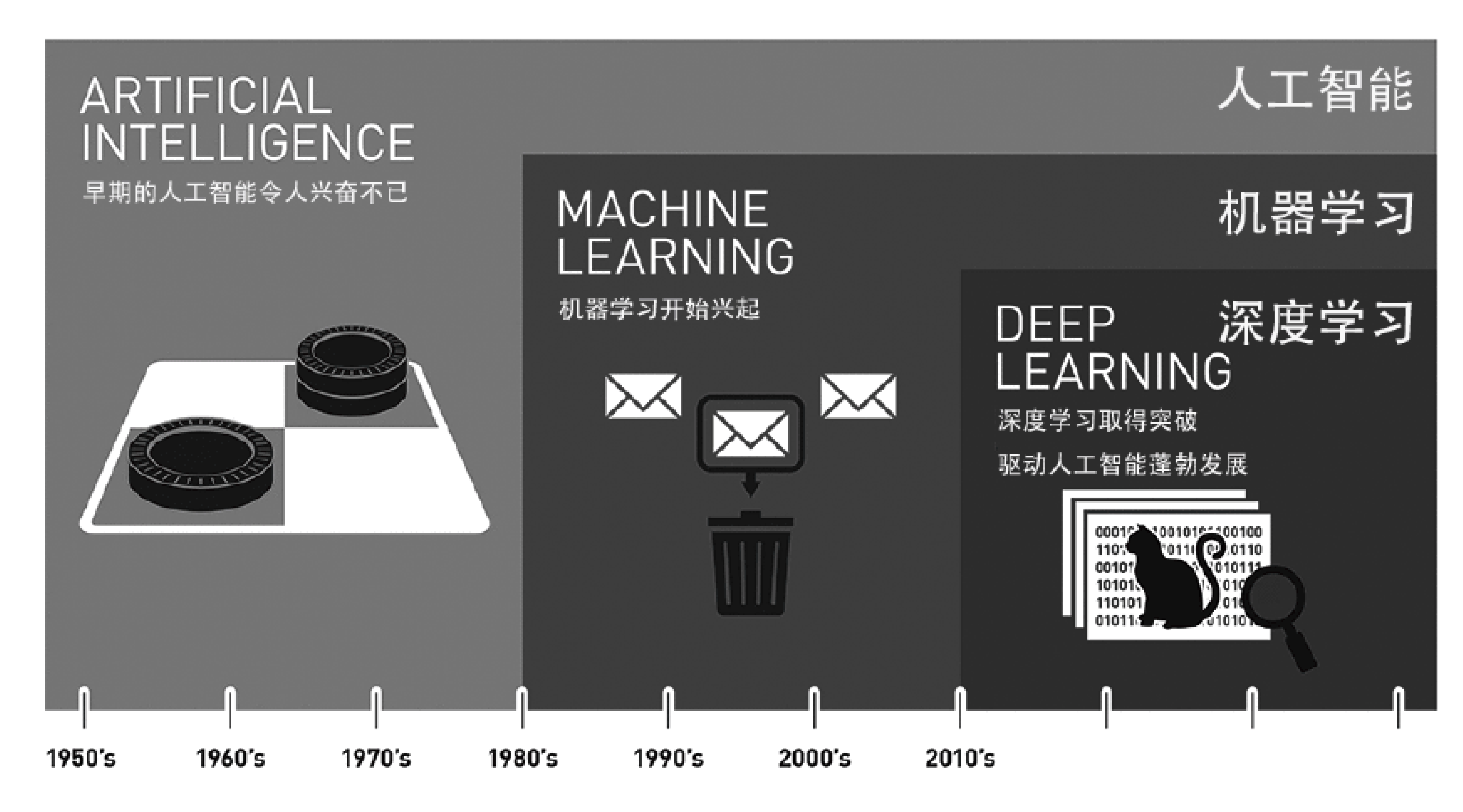


图11-5 人工智能、机器学习、深度学习的发展进程

在国内，2011年科大讯飞首次将DNN技术运用到语音云平台，并提供给开发者使用，并在讯飞语音输入法和讯飞口讯等产品中得到应用。百度成立了IDL(深度学习研究院)，专门研究深度学习算法，目前已有多项深度学习技术在百度产品上线。此外，国内其他公司，如搜狗、云知声等，也纷纷开始在产品中使用深度学习技术。

11.3.2 知识计算

要对数据进行高端分析，就需要从大数据中先抽取出有价值的知识，并把它构建成可支持查询、分析和计算的知识库。基于大数据的知识计算是大数据分析的基础，也是近年来国内外工业界、学术界研究的一个热点。目前，世界各国的相关组织建立的知识库多达50余种，相关的应用系统更是达到上百种。其中具有代表性的知识库或应用系统有Know It All、Text Runner、NELL、Probase、Satori、PROSPERA、SOFIE，以及一些基于维基百科等在线百科知识构建的知识库，如DBpedia、YAGO、Omega和Wiki Taxonomy等。

知识图谱(knowledge graph)也称为科学知识图谱，它通过将应用数学、图形学、信息可视化技术、信息科学等理论和方法与计量学引文分析、共现分析等方法结合，并利用可视化的图谱形象地展示了学科的核心结构、发展历史、前沿领域以及整体知识架构达到多学科融合目的的现代理论，是知识计算的核心。随着互联网中用户生成内容(UGC)和开放链接数据(LOD)等大量RDF数据被发布，互联网又逐步从仅包含网页与网页之间超链接的文档万维网转变为包含大量描述各种实体和实体之间丰富关系的数据万维网。在此背景下，知识图谱正式被Google公司于2012年5月提出，其目标在于改善搜索结果，描述真实世界中存在的各种实体和概念以及这些实体、概念之间的关联关系。紧随其后，国内外的其他互联网搜索引擎公司也纷纷构建了自己的知识图谱，如微软的Probase、搜狗的知立方、百度的知心。知识图谱在语义搜索、智能问答、数据挖掘、数字图书馆、推荐

系统等领域有着广泛的应用。国内学术界也对中文知识图谱的构建与知识计算进行了大量的研究和开发工作,代表性工作有中国科学院计算技术研究所的 Open KN,中国科学院数学与系统科学研究院陆汝钤院士提出的知件(knowware),百度推出的中文知识图谱搜索,搜狗推出的知立方平台,复旦大学 GDM 实验室推出的中文知识图谱展示平台(CN-DBpedia)等。

支持知识计算的基础是构建知识库,包括 3 个部分:

(1) 知识库的构建。构建几个基本的构成要素,包括抽取概念、实例、属性和关系。

- 手动构建。依靠专家知识编写一定的规则,从不同的来源收集相关的知识信息,构建知识的体系结构,例如知网(Hownet)、同义词词林、OpenCyc 等。
- 自动构建。基于知识工程、机器学习、人工智能等理论自动从互联网上采集并抽取概念、实例、属性和关系,例如 Probase、YAGO 等。

(2) 多源知识的融合。是指为了解决知识的复用问题,需要对多个来源的知识进行清理和融合工作,包括对概念、实例进行映射、消歧,对关系进行合并等。

- 手动融合。非常费时而且容易出错,一般适用规模比较小的知识库。
- 自动融合。基于机器学习、人工智能和本体工程等算法进行融合,扩展性较好。例如,YAGO 将维基百科的分类体系和 WordNet 的分类体系进行融合,Probase 基于概率化的实体消解的知识整合技术将 IMDB、Freebase 等结构化数据整合起来。

(3) 知识库的更新。知识库的更新分为两个方面,一是新知识的加入,二是已有知识的更改。从更新方式来讲分为两类:

- 基于知识库构建人员的更新。准确性较高,对人力的消耗较大。
- 基于知识库存储的时空信息的更新。由知识库自身更新,需要人工干预的较少,准确率不高。

11.3.3 社会计算

社会计算的概念首次出现于 1994 年。Schuler 指出"社会计算可以是任何一种类型的计算应用,以软件作为社交关系的媒介或聚焦",强调了社会软件应用的重要性。Dryer 等人认为:社会计算是人、社会行为及系统交互使用计算技术来相互影响,其设计模型重点分析了移动计算系统中系统设计、人类行为、社会贡献及交互结果等因素的相互作用。Charron 等人将社会计算定义为技术影响个体或社区,而非机构的社会架构。中国科学院自动化研究所王飞跃研究员从广义和狭义两个层面给出了社会计算的定义:广义而言,社会计算是指面向社会科学的计算理论和方法;狭义而言,社会计算是面向社会活动、社会过程、社会结构、社会组织及其作用和效应的计算理论和方法。

以微信、微博等为代表的在线社交网络和社会媒体正深刻改变着人们传播信息和获取信息的方式,人和人之间结成的关系网络承载着网络信息的传播,人的互联成为信息互联的载体和信息传播的媒介,社会媒体的强交互性、时效性等特点使其在信息的产生、消费和传播过程中发挥着越来越重要的作用,成为一类重要信息载体。

1. 在线社会网络的结构分析

在线社会网络在微观层面上具有随机化无序的现象,在宏观层面上往往呈现出规则化、有序的现象,社区结构作为探索和分析连接微观和宏观的网络中观结构,在近年来成为研究热点,为理清网络具有的这种看似矛盾的不同尺度的结构特性提供了很好的解决方法。社区分析研究主要包括社区的定义和度量、社区结构发现和社区结构演化性分析等基本问题。

2. 在线社会网络的信息传播模型

信息传播模型的研究包括传染病模型、随机游走模型等。传染病模型获得了广泛深入的研究和关注,然而,近年的研究也逐渐发现信息传播和传染病传播具有显著不同的特征,包括信息传播的记忆性、社会增强效应、不同传播者的角色不同、消息内容的影响等。随机游走模型则与反应-扩张过程、社团挖掘、路由选择、目标搜索等动力学过程紧密相关。当前,对在线社交网络中信息传播的研究主要集中在实证分析和统计建模,对于信息传播机理仍然缺乏深入的理解和有效的建模。

11.3.4 特异群组挖掘

存在这样一类数据挖掘需求:将大数据集中的少部分具有相似性的对象划分到若干个组中,而大部分数据对象不在任何组中,也不和其他对象相似,如图 11-6 所示。将这样的群组称为特异群组,实现这一挖掘需求的数据挖掘任务被称为特异群组挖掘(cohesive anomaly mining),由朱扬勇和熊赟于 2009 年首次提出。大数据特异群组挖掘具有广泛应用背景,在证券交易、智能交通、社会保险、生物医疗、银行金融和网络社区等领域都有应用需求,对发挥大数据在诸多领域的应用价值具有重要意义。

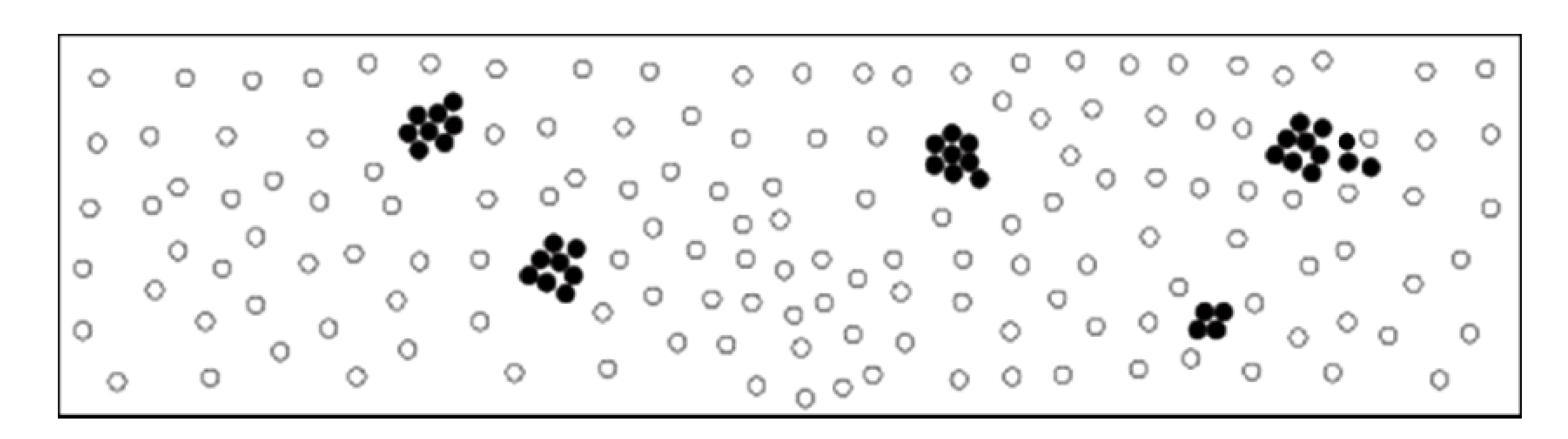


图 11-6 数据中的特异群组

1. 与聚类比较

聚类是根据最大化簇内相似性、最小化簇间相似性的原则,将数据对象集合划分成若干个簇的过程。特异群组挖掘是在大数据集中发现特异群组,找出的是少部分具有相似性的数据对象。与聚类的共同之处是,特异群组中的对象也具有相似性,并将相似对象划分到若干个组中,这在一定程度上符合传统簇的概念。但是,特异群组之外的对象数目一般远大于特异群组中对象的数目,并且这些对象不属于任何簇,这和聚类的目的是不同的。

2. 与异常检测比较

少部分数据对象的挖掘通常被认为是异常检测任务。然而,异常检测算法不能直接用来发现特异群组。一是目标不同,异常挖掘算法的目标一般是发现数据集中那些少数不属于任何簇,也不和其他对象相似的异常点;二是存在聚类假设,除异常点检测外,存在一些算法用于发现异常点成簇的情况,称为微簇挖掘,即微簇问题在一个数据集中包含点异常、微簇和簇;三是对数据集结构关系的探索要求不同,集体异常挖掘任务也不同于特异群组挖掘,因为集体异常只能出现在数据对象具有相关性的数据集中,其挖掘要求探索数据集中的结构关系,主要用于处理序列数据、图数据和空间数据等。

通过上述比较分析可以得到,挖掘的需求决定了采用哪种技术:如果需要找大部分数据对象相似,则采用聚类;需要找少部分数据对象相似,则采用特异群组挖掘;如果是找少数不相似的数据对象,则采用异常检测。

11.4 未来发展趋势

中国大数据产业发展历程如图 11-7 所示。

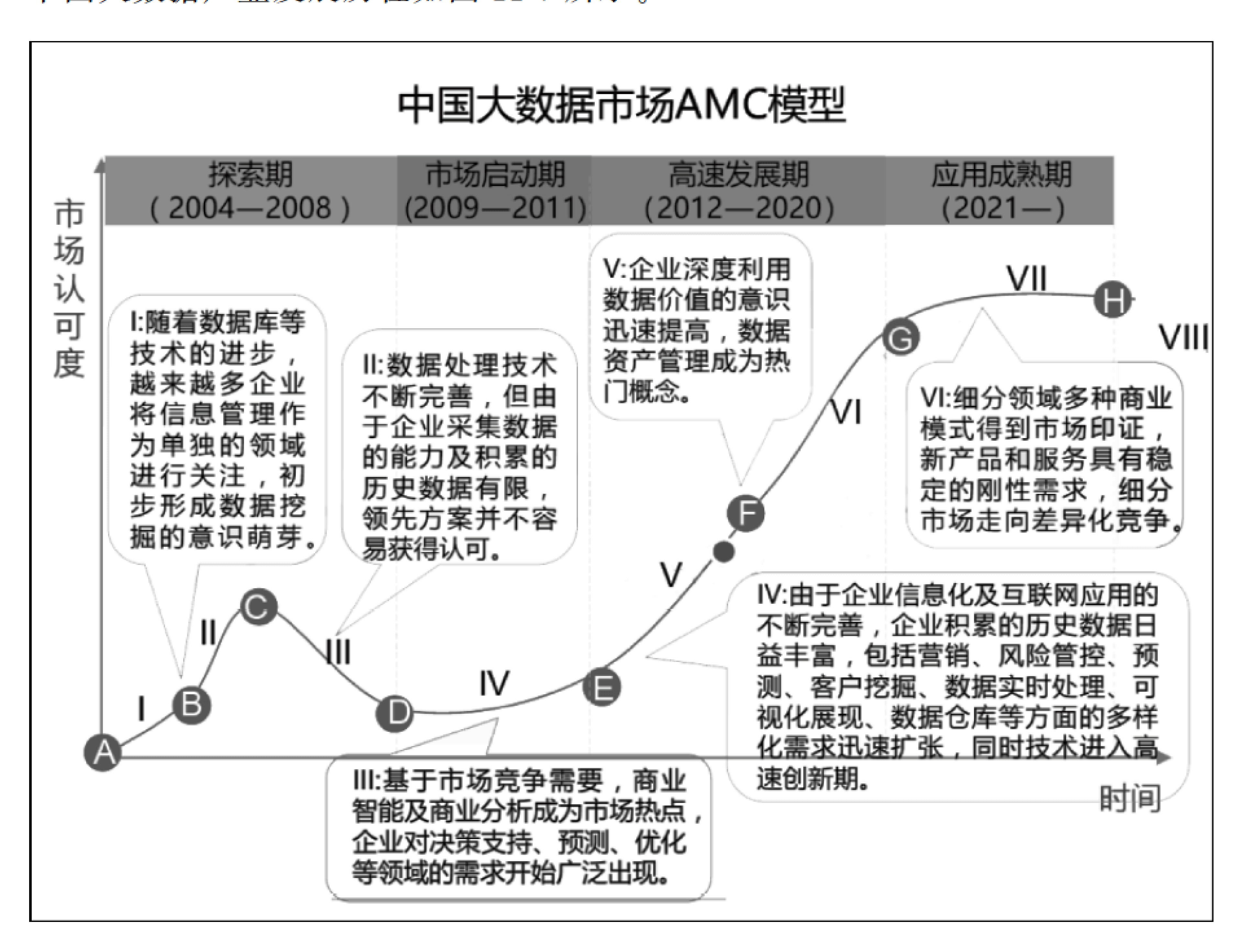


图 11-7　大数据产业发展历程

来源:《中国大数据市场年度综合报告 2016》,易观(http://www.analysys.cn)

中华人民共和国国民经济和社会发展第十三个五年规划纲要(简称"十三五"规划

(2016—2020年))中提出:"实施国家大数据战略,推进数据资源开放共享"。作为"'十三五'十四大战略"之一的"国家大数据战略",我国《大数据产业"十三五"发展规划》也正在紧张制定中。"十三五"期间,大数据领域必将迎来建设高峰和投资良机。2016年,"数据中国"建设蓄势待发,中国计算机学会大数据专家委员会发布了大数据十大发展趋势预测结论。

一是可视化推动大数据平民化。由于可视化技术的发展和广泛应用,一般用户也能够通过可视化技术来对大数据进行简单的分析处理和应用。

二是多学科融合与数据科学的兴起。大数据将促使多学科领域的进一步交叉融合发展。正如本书引言中所说的,数据科学逐渐成为一个新的学科方向。

三是大数据安全与隐私令人忧虑。在促进社会进步发展的同时,大数据将带来网络信息安全问题,需要引起高度重视。

四是新热点融入大数据多样化处理模式。大数据的处理模式不断丰富,新旧手段不断融合,例如流数据、内存计算等将成为新的热点。

五是大数据提升社会治理和民生领域应用。基于大数据的社会治理将涉及智慧城市、应急、税收、反恐、农业等多个民生领域。

六是《促进大数据发展行动纲要》驱动产业生态。大数据已经成为驱动经济转型的新动力、重塑国家竞争优势的新机遇、提升政府治理能力的新途径。

七是深度分析推进大数据智能应用。在学术技术方面深度分析将会继续推动整个大数据智能的发展。

八是数据权属与数据主权备受关注。大数据问题,从个人和一般机构层面来看,是数据权属问题;从国家层面来看,则是数据主权问题。数据的资源化和价值化也是数据权属问题和数据主权问题的根源。

九是互联网、金融、健康将继续保持热度。智慧城市、企业数据化、工业大数据将成为新的增长点。

十是开源、测评、大赛催生良性的人才与技术生态。大数据是应用驱动、技术发力,开源系统将成为大数据领域的主流技术和系统选择。以Hadoop为代表的开源技术拉开了大数据的大幕,大数据应用的发展又促进了开源技术的进一步发展,对数据处理的能力和性能等进行测试评估、标杆比对的第三方形态将会出现,并成为热点。各类创业创新比赛的举办为人才的选拔提供了新模式,完善了人才生态。2016年1月,国家发改委启动了促进大数据发展重大工程,在社会治理、公共服务、产业发展、创业创新等方面全面开展大数据示范应用工程,推进大数据共享开放,统筹发展大数据基础设施,建立完善大数据的流通交易体系。

11.5 小结

当前,大数据已成为继物联网、云计算之后的信息技术产业中最受关注的热点领域之一。人类社会已经步入了信息技术、云计算、大数据的时代,随着大数据从概念渗透转向应用发展,大数据产业正处在蓬勃发展的孕育期与机遇期,大数据技术将在开源环境下不

断提升。本书希望能够将读者引入大数据技术的大门，本章对未来的发展趋势和应用领域作了简要介绍和分析，旨在为读者进一步学习奠定基础。

11.6 参考文献

[1]《国家发展改革委办公厅关于组织实施促进大数据发展重大工程的通知》发改办高技[2016]42 号
[2] Bryant R, Katz R, Lazowska E. Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science, and Society, White Paper, Computing Community Consortium, 2008: 1-7.
[3] 联合国全球脉动项目. http://www.unglobalpulse.org/.
[4]《自然》http://www.nature.com/nature/journal/v455/n7209/full/455001a.html.
[5]《科学》http://www.sciencemag.org/site/special/data/.
[6] 美国白宫. Big Data is a Big Deal. https://www.whitehouse.gov/blog/2012/03/29/big-data-big-deal.
[7] Spark 峰会. https://spark-summit.org/2015.
[8] 任磊，杜一，等. 大数据可视分析综述[J]. 软件学报，2014，25(9): 1909-1936.
[9] 袁晓如. 大数据可视分析. 第一届科学大数据大会，北京，2014.
[10] 陶雪娇，胡晓峰，等. 大数据研究综述[J]，系统仿真学报，2013(S1): 142-146.
[11] 熊赟，朱扬勇. 特异群组挖掘: 框架与应用[J]. 大数据，2015(2): 66-77.
[12] 程学旗，靳小龙，等. 大数据系统和分析技术综述[J]. 软件学报，2014，25(9): 1889-1908.